Universidad de Alcalá
Escuela Técnica Superior de Ingeniería Informática

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

Aplicativo de Bases de Datos para Entornos Docentes

ESCUELA POLITECNICA SUPERIOR

Autor: Javier Alonso Fernández

Tutor: Manuel Sánchez Rubio

2015

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

Aplicativo de Bases de Datos para Entornos Docentes

Autor: Javier Alonso Fernández

Tutor: Manuel Sánchez Rubio

Tribunal

Presidente:

Vocal 1º:

Vocal 2º:

Alcalá de Henares a, de del 2015

A mi madre, a mi padre y a mi hermano por estar siempre ahí y facilitarme siempre las cosas por muy complicadas que las ponga.

A Mary por ser mi hermana de selección, mi segunda madre y la compañera de este viaje que no habría sido posible terminar sin ella.

A Adri por todas las tardes de arreglar el mundo sin importar el lugar, el problema o el tiempo disponible. Eres mi pata insustituible.

A Jorge y a Luke porque las tardes de montaditos y las noches frikis que son un auténtico balón anti estrés. Encima, son mejores como amigos.

A los del barrio, porque he crecido con vosotros y soy quien soy en buena parte por vosotros, y no puedo estar más orgulloso de ello.

A mi medio limón, Rocío, que es mi conciencia para el trabajo consiguiendo que la excelencia sea un requisito más a cumplir desde la sencillez. Este TFG lleva un gran pedazo de ti y lo sabes.

A Irene, por demostrar que lo que viene por detrás no son solo mejores y hay que seguir mejorando sino que encima son grandes personas.

A Gemma, por alegrar las noches de este viaje, gracias pollo.

A Rosa, por hacerme sentir que siempre podía lograrlo y que el tiempo siempre se puede estirar un poco más. Hakuna Matata, Dory.

A los miembros de All-Star porque me han aguantado asignatura tras asignatura y gran parte de mis calificaciones son suyas.

A mi familia Indrera, en especial a David, Lydia, Estefy, Miguel, Laura, Dani Co, Dani T, Fran, Diana y Cohete que consiguen que el trabajo sea más llevadero y que salir de allí y seguir aprendiendo no sea tan duro.

A mi familia Crossfitera que no hacen más que demostrarme día a día que los límites están para romperlos y que no existen edades ni estereotipos ni nada salvo lo que establezcamos nosotros mismos.

A todos aquellos que me han aportado algo a lo largo de mi vida y me han hecho llegar hasta aquí, una pieza del puzle que soy es vuestra.

Por último, y no menos importante, a Manuel Sánchez Rubio por todas las facilidades que me ha dado para la realización de este TFG, por ser un excelente profesor, haciendo que todo parezca sencillo y amistoso, muchos profesores deberían seguir su ejemplo.

Lo mejor ni ha pasado ni está por venir, está pasando.

David Martínez

Índice general

Capítulo 1
1 Resumen

1.1 Resumen

El objetivo principal del trabajo es realizar una aplicación web multiplataforma que permita realizar la gestión y la administración de las actividades extraescolares de un centro docente permitiendo llevar a cabo altas, bajas y modificaciones tanto de alumnos como de actividades.

Dicha gestión se realizará utilizando un sistema de inscripciones en las actividades mediante reservas y colas de espera que permita posteriormente realizar listados de alumnado, inscripción, reservas, cursos, actividades, etc. Para ello se va a utilizar tecnologías web como HTML5, JSP, AJAX o JavaScript corriendo bajo un servidor web Apache Tomcat junto con una base de datos MySQL.

1.2 Summary

The primary goal of this project is to develop a multiplatform Web Application with full capability of managing and administrating extracurricular activities from an elementary school. The application is able to manage new signups, rejections or modifications both of students or activities.

The managing of the activities is done following a subscription method based on reservations and queueing that provides the possibility of generate full listings of student body, signatures, reservations, courses, activities and so on. The application is developed using web technologies like HTML5, JSP, AJAX or JavaScript and it is running under an Apache Tomcat web server with a MySQL database.

1.3 Palabras Clave

JSP, HTML5, MySQL, Educación, Extraescolar.

1.4 Resumen Extendido

Es cada vez más habitual en los centros docentes, sobre todo en aquellos donde el alumnado es muy joven (hasta 12 años), que se impartan actividades extraescolares que permitan al alumnado ampliar su formación académica así como permitir también a los padres poder determinar un horario para sus hijos que sea compatible con el que ellos tienen en sus respectivos trabajos.

Por lo general estas actividades pueden ser de diversos tipos ya sean deportes, actividades culturales, idiomas, excursiones, etc. También suele darse el caso que estas actividades estén formadas por grupos reducidos de alumnos y esto conlleva a que se sucedan situaciones en muchas ocasiones en las que, debido a la alta demanda de una actividad, sea necesaria una gestión de reservas e inscripciones por parte del centro de docente para determinar qué alumnos acceden a dicha actividad.

Esta situación sumada al hecho de que los centros docentes tienen en su haber matriculados a un número muy elevado de alumnos deriva en que resulta prácticamente imposible realizar una labor de organización de las actividades extraescolares eficiente sin utilizar el apoyo de una herramienta informática como un fichero Excel o una base de datos que permita llevar una relación de las actividades extraescolares con los alumnos que han solicitado su inscripción en las mismas.

Más aún necesaria sería el uso de una herramienta informática si se desea además poder obtener listados de inscripción, reserva, pago, asistencia, etc. O incluso llevar un registro histórico o estadístico a lo largo de los años.

Dada esta situación surge este proyecto de aplicación web junto a una base de datos para proveer de una solución genérica pero a la vez con capacidad de adaptación a las necesidades de cada centro. Para el desarrollo de la aplicación se siguió el modelo de desarrollo de software de ciclo de vida en cascada y contó con las siguientes fases:

- Análisis de requisitos.
- Diseño.
- Codificación.
- Pruebas y verificación
- Mantenimiento

Cada una de las fases se fue realimentando tanto con las anteriores como con las sucesivas.

La idea de desarrollarlo como una aplicación web derivó de que actualmente vivimos en un mundo en el que todo está conectado y si bien es cierto que durante la investigación de requisitos no

todos los centros señalaron la necesidad de una aplicación online, es fácil deducir que es muy posible que en un futuro no muy lejano sí que lo requieran.

Otro motivo a favor es que una aplicación web permite utilizar el patrón de software Modelo-Vista-Controlador (MVC en adelante) que proporciona una separación de los datos y la lógica de negocio respecto de la interfaz de usuario y el modulo para gestionar los eventos y comunicaciones. MVC propone tres componentes: el Modelo, la Vista y el Controlador. Gracias a estos tres componentes es posible tener en un lado la definición de cómo se va a representar la información al usuario y por otro lado definir cómo se procesa y trata la información. Este patrón maneja dos ideas principales: la reutilización de código y la separación de conceptos.

Una vez establecida la idea a desarrollar y el cómo se va a desarrollar el siguiente paso fue acotar una serie de requisitos y objetivos a cumplir que permitieran determinar que la solución generada establecía el éxito de nuestra solución frente al problema hallado. Para ello, y más en un mundo tan cambiante año a año como es el estudiantil, lo mejor fue recabar información de primera mano acudiendo a un centro docente preguntando como realizan ellos la gestión de las actividades extraescolares, qué necesidades tienen, etc. Una vez realizada una reunión con personal del centro docente se establecieron una serie de necesidades bastante claras y que son señaladas brevemente a continuación:

Los encargados de llevar la gestión de las actividades extraescolares es la AMPA (Asociación de Madres y Padres de Alumnos). Si se está inscrito en el AMPA se tiene prioridad para acceder a las actividades. Para las actividades, se gestionan año a año generando nuevas clases y para inscribirse se realiza un período de inscripción en donde el primero en llegar es el primero en entrar y así hasta cubrir el cupo a partir del cual se entra en período de espera por medio de una cola de espera.

El AMPA guarda un registro de todas las familias con sus datos de contacto y sus miembros y al principio de cada curso genera un listado con las actividades disponibles y posteriormente con los listados de las clases y los alumnos admitidos que se han creado.

A partir de esta información se pueden establecer los siguientes requisitos:

Requisitos funcionales:

- Altas y modificaciones de actividades.
- Altas, modificaciones y borrados de clases de las actividades.
- Altas y modificaciones de alumnos y de sus familias.
- Alta de alumnos en actividades (inscripciones) y su gestión por cola de espera.
- Capacidad para generar informes de actividades.
- Capacidad para generar informes de alumnos y familiares.
- Capacidad para generar informes de alumnos inscritos en actividades.
- Control de acceso al sistema por identificación.

A estos requisitos generales se estimó conveniente añadir los siguientes debido a su capacidad de mejorar y añadir más funcionalidades lo que, por ende, supuso añadir más valor a la aplicación:

- Envío de circulares mediante correo electrónico.
- Búsqueda de alumnos.
- Búsqueda de actividades.

Fue también necesario establecer una serie de requisitos no funcionales que terminaron de moldear la base necesaria de la aplicación. Estos requisitos giran en torno a materias como la portabilidad de la aplicación, disponibilidad, etc.

- La aplicación debe ser capaz de estar disponible las 24 horas del día durante todos los días del año.
- Se utilizará el patrón de diseño Modelo-Vista-Controlador.
- La aplicación debe ser capaz de adaptarse a entornos móviles.

Finalizado el establecimiento de requisitos, era hora de comenzar el desarrollo de la aplicación. Lo primero fue establecer unos casos de uso que permitieran trazar todos los requisitos establecidos de una manera sencilla y eficaz y que además resultase intuitiva al usuario final. En total se han generado un total de 18 casos de uso.

Tras la creación de los casos de uso se extrajeron las entidades más significativas obteniéndose 6: Familia, DatoContacto, Persona, Actividad, Clase y Horario. A partir de aquí se establecieron los diccionarios de datos para cada una de ellas y se trasladó todo a un modelo entidad relación para poder plasmar toda la información en MySQL y establecer así finalmente la capa lógica de datos de la aplicación y en lo que se basaría toda la capa modelo de MVC. El modelo entidad relación dio como resultado 7 tablas a crear en la base de datos: Familia, DatoContacto, Persona, Actividad, Clase y Horario e Inscripciones que surgió de la relación entre Clase y Persona.

Una vez establecida la base de la parte del modelo se continuó desarrollando las clases JAVA que representasen las tablas de la base de datos y posteriormente se establecieron las clases que harían de interfaces de comunicación creando 7 DAOs para poder realizar las operaciones de inserción, borrado, edición y rescate de datos. Una vez desarrollada toda esta parte se pudo dar por cerrada la capa Modelo de MVC.

Se continuó entonces con la capa Vista de MVC creando las páginas JSP en conjunto con Bootstrap. Primero se generó una primera versión sin nada de JSP y tan sólo con HTML5 y Bootstrap para establecer un borrador de cómo se deseaba que fueran las páginas. Tras esto se fue añadiendo poco a poco JSP e inteligencia a las páginas con jQuery y jqGrid. Esto dio por cerrada la parte de Vista de MVC.

Tras esto se pasó a interconectar ambas campas, vista y modelo, mediante la creación de la capa controlador. Para ello se desarrollaron los Servlet que escuchan las peticiones que realiza el usuario en las páginas JSP y que reclaman a la base de datos operaciones. Primero se realizaron servlets que hicieran operaciones de rescate de datos a la base de datos para poder ver que estos se recogían bien y visualizaban correctamente. Así se consiguió depurar todas las capas a la vez.

Finalmente se continuó con los servlets que añadían datos a la base de datos para posteriormente realizar los de modificación y los de borrado. Finalizado este proceso la aplicación estaba ya plenamente desarrollada en sus funcionalidades más críticas y se pasó entonces a las secundarias. Fue entonces cuando se desarrolló la parte de generar los listados en PDF, exportar a Excel las tablas jqGrid, el envío de email, etc.

Como último paso, con toda la aplicación creada se procedió a realizar pruebas y más pruebas que comprobasen que todo funcionase correctamente y se procedió a la redacción de esta memoria.

Capítulo 2
2 Memoria

2.1 Introducción

Es cada vez más habitual en los centros docentes, sobre todo en aquellos donde el alumnado es muy joven (hasta 12 años), que se impartan actividades extraescolares que permitan al alumnado ampliar su formación académica así como permitir también a los padres poder determinar un horario para sus hijos que sea compatible con el que ellos tienen en sus respectivos trabajos.

Por lo general estas actividades pueden ser de diversos tipos ya sean deportes, actividades culturales, idiomas, excursiones, etc. También es suele darse el caso que estas actividades estén formadas por grupos reducidos de alumnos y esto conlleva a que se sucedan situaciones en muchas ocasiones en las que, debido a la alta demanda de una actividad, sea necesaria una gestión de reservas e inscripciones por parte del centro de docente para determinar qué alumnos acceden a dicha actividad.

Esta situación sumada al hecho de que los centros docentes tienen en su haber matriculados a un número muy elevado de alumnos deriva en que resulta prácticamente imposible realizar una labor de organización de las actividades extraescolares eficiente sin utilizar el apoyo de una herramienta informática como un fichero Excel o una base de datos que permita llevar una relación de las actividades extraescolares con los alumnos que han solicitado su inscripción en las mismas.

Más aún necesaria sería el uso de una herramienta informática si se desea además poder obtener listados de inscripción, reserva, pago, asistencia, etc. O incluso llevar un registro histórico o estadístico a lo largo de los años.

Dada esta situación surge este proyecto de aplicación web junto a una base de datos para proveer de una solución genérica pero a la vez con capacidad de adaptación a las necesidades de cada centro.

2.2 Base teórica

2.2.1 Modelo de desarrollo en cascada

El presente proyecto ha sido desarrollado siguiendo el modelo en cascada que sigue un enfoque metodológico que ordena cada una de las etapas del proceso para el desarrollo de software, de tal forma que el comienzo de las mismas debe esperar a la finalización de la etapa anterior ya que para finalizar una etapa es necesario realizar una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado lo que conlleva un aumento en los costos de desarrollo. Pese a que este modelo ha sido criticado en numerosas ocasionas, hoy en día sigue siendo el más utilizado a día de hoy. Por lo general todos los proyectos cuentan con las siguientes fases o etapas:

- **Análisis de requisitos**: En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. Es importante señalar que en esta etapa se debe fijar claramente todo lo que se requiere del sistema ya que supondrá la base y guía en las siguientes etapas, no pudiéndose requerir nuevos requisitos a mitad del proceso de elaboración del software que supongan cambios radicales en la idea del proyecto.
- **Diseño**: Esta fase supone la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras. Aquí se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber qué herramientas usar en la etapa de codificación.
- **Codificación**: Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores.
- **Pruebas y verificación**: Una vez logrado tener todos los elementos programados, se ensamblan para componer el sistema o aplicación completo y se comprueba que funciona correctamente y que cumple con los requisitos, antes de ser entregado al usuario final para que realice la verificación final.
- **Mantenimiento**: Una de las etapas más críticas, ya que se destina hasta un 75 % de los recursos. El mantenimiento del Software supone las modificaciones y actualizaciones solicitadas por el cliente final derivadas de su uso a lo largo del tiempo o la detección de nuevas necesidades o errores.

2.2.2 Modelo Vista Controlador

El modelo-vista-controlador (en adelante MVC) es un patrón de arquitectura de software que separa en una aplicación los datos y la lógica de negocio de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código, utilizar una parte de un programa en la creación de distintas aplicaciones, y la separación de conceptos, dividir un programa en pequeños módulos encapsulados en interfaces que se implementan en cada programa de una manera distinta si así se desea. Estas dos características buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. Los tres componentes se definen de una manera genérica:

- **Modelo**: Es la representación de la información con la cual el sistema o aplicación opera, por lo tanto se encarga de gestionar todos los accesos a dicha información, es decir, consultas, actualizaciones e incluso implementa también los privilegios de acceso que se hayan descrito como requisito de la aplicación (lógica de negocio). Se comunica con la 'vista' enviándole aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador' que las habrá recibido previamente normalmente por el usuario de la aplicación.
- **Controlador**: Es el componente responsable de responder a eventos generados, por lo general por el usuario, en la aplicación. A raíz de estos eventos, invoca peticiones al 'modelo' cuando se hace alguna solicitud de información, por ejemplo, dar de alta a un alumno en la base de datos. También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo', por ejemplo, recorrer un listado de alumnos en una tabla que se presenta al usuario. Por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.
- **Vista**: Presenta el 'modelo', la información y lógica de negocio, en un formato adecuado para interactuar con ella, normalmente una interfaz de usuario (en nuestro caso, la página web donde se encuentre); por tanto, requiere de dicho 'modelo' la información que debe representar como salida.

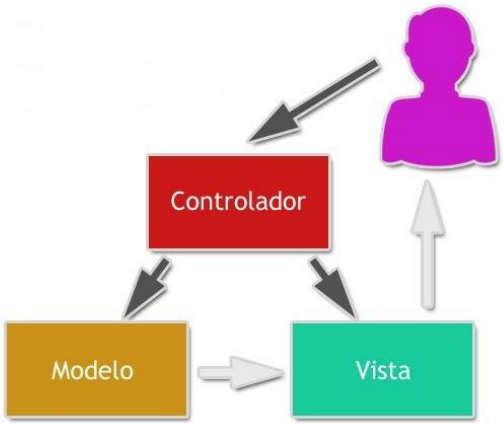

Ilustración 1 Interacción componentes MVC

Un flujo de control típico de MVC sería:

1. El usuario interactúa con la interfaz de usuario de alguna forma, por ejemplo, el usuario pulsa un enlace.
2. El controlador recibe, por parte de los objetos de la interfaz-vista, la ocurrencia del evento que ha disparado el usuario. El controlador gestiona el evento que llega, habitualmente, a través de un gestor de eventos conocido como handler o callback. En este proyecto los handlers serían los servlets que se explicarían más adelante.
3. El controlador accede al modelo, actualizándolo, consultándolo o cualquiera que sea la acción solicitada por el usuario. Un ejemplo sería borrar un registro de la base de datos. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos para generar la interfaz del modelo. El modelo no debe tener conocimiento directo sobre la vista.
5. El usuario recupera el control de la interfaz del usuario y genera nuevos eventos disparando de nuevo el ciclo explicado.

En aplicaciones como la de este proyecto donde se utiliza un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocida en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos. MVC no pretende discriminar entre capa de negocio y capa de presentación pero sí pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y el posterior mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

2.2.3 Servlets

Servlet es una clase definida dentro del lenguaje de programación Java y es utilizada para ampliar las capacidades de las que dispone un servidor. Un objeto de la clase servlet tiene la capacidad de responder a cualquier tipo de solicitud aunque por lo general los objetos servlets son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java pero que se ejecutan en servidores en vez de navegadores web. Este tipo de servlets son la contraparte Java de otras tecnologías de contenido dinámico Web, como PHP y ASP.NET. La palabra servlet deriva de otra anterior, applet, que se refiere a pequeños programas que se ejecutan en el contexto de un navegador web.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Cuando es utilizado un servlet tiene un ciclo de vida característico

- **Inicialización del servlet**: Cuando un servidor carga un servlet, ejecuta el método init del servlet. Este proceso de inicialización debe completarse antes de poder atender y manejar las peticiones de los clientes, y antes de que el servlet pueda ser destruido. Si un servlet es ejecutado en servidores multi hilo, no es necesario llamar al método init cada vez que se cree un hilo de atención ya que los servlets no tienen problemas de concurrencia durante su inicialización. Por tanto, basta con que el servidor llame sólo una vez al método init al crear la instancia del servlet, y no será necesario que invoque este método de nuevo a menos que vuelva a recargar el servlet porque éste haya sufrido cambios. El servidor no puede recargar un servlet sin primero haber destruido el servlet llamando al método destroy.
- **Interactuar con los clientes**: Una vez el servlet ha completado su inicialización, el servlet puede comenzar a atender y procesar todas las peticiones de los clientes. Estas peticiones serán atendidas por la misma instancia del servlet, por lo que hay que tener cuidado al acceder a variables compartidas, ya que podrían darse problemas de sincronización entre requerimientos simultáneos.
- **Destruir el servlet**: Los servlets se ejecutan hasta que el servidor los destruye. Los motivos para destruir un servlet son dos, por cierre del servidor o bien por petición del administrador del sistema. Cuando un servidor destruye un servlet, ejecuta el método destroy del propio servlet. Este método sólo se ejecuta una vez independientemente de que haya un servlet multihilo y puede ser llamado cuando aún queden respuestas en proceso, por lo que hay que tener cuidado y no invocar el método hasta que hayan sido atendidas todas las peticiones pendientes. El servidor no volverá a ejecutar de nuevo el servlet hasta haberlo cargado e inicializado de nuevo.

2.2.4 Java Server Pages

Java Server Pages (en adelante JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas tanto en HTML como en XML entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java apoyándose sobre todo en la clase Servlet. Es por esto que es necesario para utilizar JSP un servidor web compatible con contenedores servlet como es Apache Tomcat que es detallado más adelante. Las páginas diseñadas en JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce en el servidor a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

Se puede afirmar que un JSP se comporta como un servlet en su rendimiento ya que el código es compilado como cualquier otra clase Java. La máquina virtual de Java compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Gracias a esto JSP obtiene un buen desempeño y resulta más eficiente que otras tecnologías web que ejecutan el código interpretándolo al completo.

También derivado de utilizar Java obtiene dos de sus grandes ventajas, su portabilidad a cualquier plataforma permitiendo con ello, por ejemplo, desarrollar una web en una plataforma para terminar utilizándola en otra distinta; y también hereda la capacidad de utilizar clases que manejen la lógica de negocio y el acceso al modelo de datos. Por lo general JSP se encargará de generar el documento HTML en el archivo JSP, es decir se ocupará de la capa de presentación.

Cada JSP se ejecuta en un hilo propio con un contexto propio pero no se ejecutar con cada petición nueva sino que persiste de una petición a la siguiente ahorrando así el paso de inicialización cada vez lo que supone un ahorro de tiempo. Esta persistencia permite también ejecutar de forma más eficiente procesos como mantener una conexión a una base de datos o manejar sesiones.

2.2.5 HTML5

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML. Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada,

como los elementos <audio> y <video>. Mejora el elemento <canvas>, capaz de renderizar elementos 3D en los navegadores más importantes (Firefox, Chrome, Opera, Safari e Internet Explorer).

Algunos elementos de HTML 4.01 han quedado obsoletos, incluyendo elementos puramente de presentación, como <font> y , cuyos efectos son manejados por Hojas de estilo en cascada. También hay un renovado énfasis en la importancia del scripting DOM para el comportamiento de la web 2.0.

Como principales novedades en esta versión cabe destacar:

- Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con codecs para mostrar los contenidos multimedia.
- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime, etc) y facilidades para validar el contenido sin Javascript o incluir información de ayuda para el usuario a la hora de rellenar los datos como la etiqueta placeholder.
- Drag & Drop. Nueva funcionalidad para arrastrar objetos como imágenes.
- Añade etiquetas para manejar la Web Semántica (Web 3.0): header, footer, article, nav, time (fecha del contenido), link rel=‘’ (tipo de contenido que se enlaza). Estas etiquetas permiten describir cuál es el significado del contenido. Por ejemplo su importancia, su finalidad y las relaciones que existen. No tienen especial impacto en la visualización, se orientan a buscadores. Los buscadores podrán indexar e interpretar esta meta información para no buscar simplemente apariciones de palabras en el texto de la página.
    - Inclusión de nuevas APIs para geolocalización (en dispositivos que lo soporten), para almacenamiento en base de datos SQLite o en dominio web (local y global storage), para sockets o para creación de hilos de ejecución en paralelo.

2.2.6 JavaScript

JavaScript es un lenguaje de programación interpretado y proviene del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se implemente principalmente en los navegadores web para controlar el lado del cliente y permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, es decir, manejar la interacción que hace con la página web y ofrecerle la mejor experiencia de uso posible antes de enviar información al servidor. JavaScript incluso puede llegar a realizar las peticiones al servidor.

Es importante destacar el hecho de que se ejecuta en el lado del cliente luego no es necesario realizar ninguna instalación en el servidor y basta con integrar su código dentro de las páginas web ya

que todos los navegadores modernos interpretan el código JavaScript. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model o Modelo de Objetos del Documento (en adelante DOM).

El uso más común de JavaScript es escribir funciones embebidas o incluidas en páginas HTML y que interactúan con el DOM de la página. Algunos ejemplos sencillos de este uso son:

- Cargar nuevo contenido de la página o enviar datos al servidor a través de AJAX sin necesidad de recargar la página (por ejemplo, recargar un listado de una tabla con nuevos datos).
- Animación de los elementos de página, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.
- Validación de los valores de entrada de un formulario web para asegurarse de que son aceptables antes de ser enviado al servidor.
- Recopilación y posterior transmisión de información sobre la interacción del usuario con la página para poder realizar análisis de la experiencia del usuario: si visita los anuncios, si realiza click en X sección, etc.

2.2.7 jQuery

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos y utilizar la técnica AJAX en páginas web.

jQuery es software libre y de código abierto, está licencia bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos tanto libres como privados. La principal funcionalidad de jQuery, es que agrupa en una gran variedad de métodos y funciones, funcionalidad que cualquiera podría desarrollar en JavaScript pero que requieren de mucho código y tiempo, es decir, con las funciones propias de jQuery se logran grandes resultados en menos tiempo y espacio.

La forma de interactuar con la página es mediante la función $(), que es un alias de la función jQuery () que recibe como parámetro una expresión CSS o el nombre de una etiqueta HTML y devuelve todos los nodos (elementos) que concuerden con la expresión.

2.2.8 jqGrid

Es un plugin de jQuery cuya función principal es la de presentar tablas con listados que permitan crear tablas dinámicas en las páginas JSP. Estas tablas podrán tener paginación y permitir diversas funcionalidades como la ordenación por uno o más campos, búsqueda de registros, adición o eliminación de los mismos y otras funcionalidades.

La información a mostrar en una tabla jqGrid se recuperará de la base de datos por medio de una petición asíncrona AJAX a un Servlet asociado en nuestra aplicación.

AJAX (Asynchronous JavaScript And XML) es una técnica que permite realizar peticiones asíncronas HTTP, tanto de tipo POST como GET a partes de nuestra aplicación, sin necesidad de recargar la página en la que el usuario está navegando. La petición es lanzada desde el propio navegador del cliente de manera asíncrona, es decir, los datos se solicitan al servidor y esto no interfiere con la carga de la página; una vez tenga los datos, se le muestran al usuario modificando el contenido del DOM HTML de forma dinámica.

Para la utilización de una tabla jqGrid en una página HTML es necesario asociar un elemento <table> del DOM con una instancia de jqGrid por medio de la función de jQuery .jqGrid, que será la encargada en primer lugar de realizar la llamada vía AJAX al servidor y posteriormente, con los resultados obtenidos de la llamada, establecer la estructura y formato del listado resultante.

Desde el punto de vista del servidor, en el Servlet se recuperará normalmente la información de la base de datos, y se mostrará la información por medio de un objeto de tipo String con formato JSON que será interpretado por el código JavaScript en el callback de la función. A continuación vemos un ejemplo ilustrativo.

```
        List<Persona> listaPersonas;

        // Recuperamos todas las familias de la bd que son
SOCIOS
        listaPersonas =
DAOPersonas.getAlumnos(Integer.parseInt(request.getParameter("ti
poListado")));

        // Establecemos el total de páginas, página actual y
demás datos
        int totalNumberOfPages = ((new
BigDecimal(listaPersonas.size()))
                .divide(new BigDecimal(10),
RoundingMode.UP)).intValue();
        int currentPageNumber = 1;
        int totalNumberOfRecords = listaPersonas.size();

        // Esta clase se encarga de preparar los datos
        JqGridData<Persona> gridData = new JqGridData<Persona>(
                totalNumberOfPages, currentPageNumber,
totalNumberOfRecords,
                listaPersonas);
        System.out.println("Grid Data: " +
gridData.getJsonString());

        request.setAttribute("tipoListado",
request.getParameter("tipoListado"));
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
        // Se pinta en la pantalla el resultado, la llamada ha
sido vía AJAX
        response.getWriter().write(gridData.getJsonString());
```

Ilustración 2 Ejemplo de listado jqGrid

De la definición de los jqGrid, destacamos los siguientes atributos:

- **url**: Indica la dirección del Servlet de donde se extraerán los datos a representar.
- **datatype**: Se establece el formato de los datos a interpretar procedentes del Servlet, en nuestro caso JSON (otro formato válido sería XML).
- **colModel**: Se especifican los nombres de los objetos a representar presentes en el JSON, así como otras características de las columnas del Grid como por ejemplo si son editables, si tienen alguna restricción para introducir datos (tiene que ser un email, número, fecha, etc.), si pueden usarse para ordenar los datos, buscar, etc.
- **colName**: Se concretan las etiquetas de los encabezados de las columnas.

Resulta interesante también destacar la posibilidad de jqGrid para realizar la exportación de sus tablas a formato Excel de una manera sencilla y rápida desde el lado del cliente, lo que supone no cargar el servidor con proceso computacional. Para realizar dicha exportación basta con incluir un botón

personalizado en la tabla con una llamada a la función que configura el fichero Excel resultante de la operación.

```
jQuery("#projectTable").setGridWidth($("#projectTable").parent
().width());

            //Botón para exportar a Excel//
            paginador =
$("#projectTable").getGridParam('pagingDiv');
            jQuery("#projectTable").navButtonAdd(paginador, {
                caption: "Exportar Excel",
                buttonicon: "ui-icon-note",
                onClickButton: function() {

$("#projectTable").jqGrid('exportarExcelCliente',

{nombre:"Informe_actividades",formato:"excel"});
                },
                position: "last"
            });
```

Ilustración 3 Código JavaScript para Exportar jqGrid a Excel

A nivel JavaScript la única dependencia indispensable para el funcionamiento de jqGrid es la librería jQuery. Sin embargo, es recomendable incluir las bibliotecas de idioma para de cara a favorecer la experiencia del usuario.

2.2.9 Bootstrap

Bootstrap, creado por Twitter, es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Incluye plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales para manejar todo este contenido. Actualmente se encuentra en su versión 3.3.4 y es de código abierto y está disponible en GitHub.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS 3, pero es compatible con la mayoría de los navegadores web. Trabaja con un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores de tal forma que en aquellos donde sea compatible dicha funcionalidad estará habilitada pero en aquellos donde no sea compatible, estará desactividado y toda la página se adaptará de manera transparente para el usuario final. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos. Esto extiende la funcionalidad de la herramienta, pero no es requerida para su uso.

Desde la versión 2.0 también soporta diseños responsive. Esto significa que el diseño gráfico de la página se ajusta dinámicamente a las características del dispositivo donde se esté visualizando la

página ya sea un ordenador con una gran resolución o un dispositivo Smartphone de pantalla y resolución pequeña. Está adaptación es continua, es decir, si por ejemplo estamos trabajando en un ordenador a pantalla completa y decidimos reducir la pantalla, Bootstrap automáticamente adaptará la visualización a estas nuevas dimensiones.

Bootstrap cuenta con una estructura y funcionamiento modular y consiste esencialmente en una serie de hojas de estilo LESS que implementan la variedad de componentes de la herramienta. Una hoja de estilo llamada bootstrap.less incluye los componentes de las hojas de estilo. Los desarrolladores pueden adaptar el mismo archivo de Bootstrap, seleccionando los componentes que deseen usar en su proyecto. Los ajustes son posibles en una medida limitada a través de una hoja de estilo de configuración central. Para cambios más específicos es necesario utilizar las declaraciones LESS.

2.2.10 Apache Tomcat

Apache Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP). Tomcat es desarrollado y actualizado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 8.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Tomcat es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

La jerarquía de directorios de instalación de Tomcat incluye:

- **Bin**: arranque, cierre, y otros scripts y ejecutables.
- **Common**: clases comunes que pueden utilizar Catalina y las aplicaciones web.
- **Conf**: ficheros XML y los correspondientes DTD para la configuración de Tomcat.
- **Logs**: logs de Catalina y de las aplicaciones.
- **Server**: clases utilizadas solamente por Catalina

- **Shared**: clases compartidas por todas las aplicaciones web.
- **Webapps**: directorio que contiene las aplicaciones web.
- **Work**: almacenamiento temporal de ficheros y directorios.

2.2.11 SMTP

El Simple Mail Transfer Protocol (SMTP) es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA, teléfonos móviles, etc.). Fue definido en el RFC 2821 y es un estándar oficial de Internet.

El funcionamiento de este protocolo se da en línea, de manera que opera en los servicios de correo electrónico. Sin embargo, este protocolo posee algunas limitaciones en cuanto a la recepción de mensajes en el servidor de destino (cola de mensajes recibidos). Como alternativa a esta limitación se asocia normalmente a este protocolo con otros, como el POP o IMAP, otorgando a SMTP la tarea específica de enviar correo, y recibirlos empleando los otros protocolos.

Los administradores de servidores pueden elegir si los clientes utilizan el puerto 25 o el puerto 587 para retransmitir el correo saliente. Las especificaciones y muchos servidores soportan ambas situaciones sin problemas e incluso algunos servidores soportan el puerto 465 para el legado SMTP seguro en violación de las especificaciones pero esta práctica no es recomendable y es preferible utilizar los puertos estándar y comandos ESMTP estándar de acuerdo con la RFC 3207, si se debe utilizar una sesión segura entre el cliente y el servidor.

Es un protocolo orientado a la conexión basado en texto, en el que un remitente de correo se comunica con un receptor de correo electrónico mediante la emisión de secuencias de comandos y el suministro de los datos necesarios en un canal de flujo de datos ordenado fiable, normalmente un protocolo de control de transmisión de conexión (TCP). Una sesión SMTP consiste en comandos originados por un cliente SMTP (el agente de inicio, emisor o transmisor) y las respuestas correspondientes del SMTP del servidor (el agente de escucha, o receptor) para que la sesión se abra y se intercambien los parámetros de la sesión. Una sesión puede incluir cero o más transacciones SMTP. Una transacción de SMTP se compone de tres secuencias de comando / respuesta que son:

- **MAIL**: comando para establecer la dirección de retorno, también conocido como Return-Path, remitente o sobre. Esta es la dirección para mensajes de despedida.
- **RCPT**: comando, para establecer un destinatario de este mensaje. Este mandato puede emitirse varias veces, una para cada destinatario.
- **DATA**: para enviar el mensaje de texto. Este es el contenido del mensaje. Se compone de una cabecera de mensaje y el cuerpo del mensaje separado por una línea en blanco. DATA es en realidad un grupo de comandos, y el servidor responde dos veces: una vez para reconocer que está listo para recibir el texto, y la segunda vez después de la secuencia final de los datos, para aceptar o rechazar todo el mensaje.

2.2.12 Dynamic Reports

Es una librería desarrollada en JAVA y distribuida de manera libre que crear informes de forma dinámica sin necesidad de un diseñador visual. Destaca porque con muy pocas líneas de código es posible crear informes de una asombrosa calidad y utilidad. Otra de sus características más interesantes es la posibilidad de diseñar estos reportes de forma ad-hoc y siguiendo unas directrices impuestas desde código JAVA; esto llevado a la utilidad de la aplicación permite la creación de informes dinámicos en función de las elecciones del usuario como por ejemplo crear una lista de los participantes de una actividad que se celebre los martes y jueves de 19h a 20h.

Otra gran cualidad de Dynamic Reports es que permite aprovechar la técnica de programación orientada a objetos de herencia, permitiendo crear diseños de informes de genéricos que posteriormente sean heredados por clases más específicas que terminen de crear el diseño final del informe.

2.2.13 MySQL

MySQL es un sistema gestor de bases de datos (SGBD en adelante) relacional, multihilo y multiusuario. Es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta colección puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder a y procesar los datos guardados en un ordenador, es necesario un SGBD MySQL Server. Debido a que los ordenadores son muy eficientes manejando grandes cantidades de información, los SGBD juegan un papel central en computación ya sea como aplicaciones independientes o como parte de otras aplicaciones, como es el caso de este proyecto, una aplicación web.

Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un único y gran fichero. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar los datos de diferentes tablas si así se solicita.

Las siguientes características son implementadas únicamente por MySQL:

- Permite escoger entre múltiples motores de almacenamiento para cada tabla. Los hay nativos como MyISAM, Falcon, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example o desarrollados por partners como solidDB, NitroEDB, ScaleDB, TokuDB, IBM.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

2.3 Descripción experimental

Esta sección muestra todas las decisiones y consideraciones realizadas durante el desarrollo de este proyecto, desde su conceptualización hasta su puesta en marcha en un entorno real pasando por su diseño, codificación, despliegue y manual de uso.

2.3.1 Requisitos

2.3.1.1.1 Usuarios de la aplicación

Tipo de Usuario	Administrador
Formación	Titulación no superior / Titulación especializada en RRHH.
Habilidades	Conocer como está organizado todo el entramado de actividades extraescolares del colegio
Actividades	Realización de todas y cada una de las funcionalidades de la aplicación.

Tabla 1 Información Usuario Administrador

2.3.1.2 Restricciones

El software se debe diseñar siguiendo el patrón modelo-vista-controlador.

Será necesario que haya un sistema de autentificación para entrar en la aplicación mediante un sistema de login más contraseña, siendo el login un email registrado previamente en la aplicación.

La aplicación debe ser capaz de adaptarse a entornos móviles.

2.3.1.3 Requisitos funcionales

Número de Requisito	**RF1**		
Nombre	**Gestión de Actividades**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Los usuarios podrán acceder a la aplicación para manejar las actividades registradas en el mismo.		

Tabla 2 RF1 – Gestión de Actividades

Número de Requisito	**RF1.1**		
Nombre	**Alta de una Actividad**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Actividades” será necesario registrar nuevas actividades donde inscribir posteriormente a personas pertenecientes a una familia.		
Precondiciones	El usuario deberá estar identificado en la aplicación y los datos introducidos deben cumplir las restricciones del modelo de datos.		

Tabla 3 RF1.1 – Alta de una Actividad

Número de Requisito	**RF1.2**		
Nombre	**Modificación de una Actividad**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Actividades” será necesario poder modificar una actividad existente.		
Precondiciones	El usuario deberá estar identificado en la aplicación, la actividad a modificar debe existir y los datos introducidos deben cumplir las restricciones del modelo de datos.		

Tabla 4 RF1.2 – Modificación de una Actividad

Número de Requisito	**RF1.3**		
Nombre	**Consulta de una Actividad**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Actividades” se dará la posibilidad al usuario de poder consultar la información referida a una actividad.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la actividad a consultar existir.		

Tabla 5 RF1.3 – Consulta de una Actividad

Número de Requisito	**RF1.4**		
Nombre	**Generar Actividades por Años**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Actividades” se dará la posibilidad al usuario de tener las actividades divididas por años.		
Precondiciones	El usuario deberá estar identificado en la aplicación.		

Tabla 6 RF1.4 – Clasificación de Actividades por Años

Número de Requisito	**RF2**		
Nombre	**Gestión de Familias**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Los usuarios podrán acceder a la aplicación para manejar las familias registradas en la misma.		

Tabla 7 RF2 – Gestión de Familias

Número de Requisito	**RF2.1**		
Nombre	**Alta de una Familia**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” será necesario registrar nuevas familias cuyos familiares serán inscritos con posterioridad en las actividades.		
Precondiciones	El usuario deberá estar identificado en la aplicación y los datos introducidos deben cumplir las restricciones del modelo de datos.		

Tabla 8 RF2.1 – Alta de una Actividad

Número de Requisito	**RF2.2**		
Nombre	**Modificación de una Familia**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” será necesario poder modificar una familia existente.		
Precondiciones	El usuario deberá estar identificado en la aplicación, la familia a modificar debe existir y los datos introducidos deben cumplir las restricciones del modelo de datos.		

Tabla 9 RF2.2 – Modificación de una Actividad

Número de Requisito	**RF2.3**		
Nombre	**Consulta de una Familia**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” se dará la posibilidad al usuario de poder consultar la información referida a una actividad.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la familia a consultar existir.		

Tabla 10 RF2.3 – Consulta de una Actividad

Número de Requisito	**RF2.4**		
Nombre	**Alta de un familiar**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” se dará la posibilidad al usuario de poder realizar el alta de un familiar nuevo para una familia ya existente en la base de datos.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la familia a la que se le va a añadir el familiar deberá existir.		

Tabla 11 RF2.4 – Alta de una familiar

Número de Requisito	**RF2.5**		
Nombre	**Modificación de un familiar**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” se dará la posibilidad al usuario de poder realizar la modificación de un familiar para una familia ya existente en la base de datos.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la familia y el familiar a modificar deberán existir. Además los datos modificados deberán cumplir las restricciones del modelo de datos.		

Tabla 12 RF2.5 – Modificación de un familiar

Número de Requisito	**RF2.6**		
Nombre	**Borrado de un familiar**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad “Gestión de Familias” se dará la posibilidad al usuario de poder realizar la eliminación de un familiar para una familia ya existente en la base de datos.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la familia y el familiar a borrar deberán existir y este no deberá ser de tipo SOCIO.		

Tabla 13 RF2.6 – Borrado de un familiar

Número de Requisito	**RF3**		
Nombre	**Gestión de Inscripciones**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Los usuarios podrán acceder la aplicación para gestionar las inscripciones de miembros de una familia en las actividades generadas en un año concreto.		

Tabla 14 RF1 – Gestión de Inscripciones

Número de Requisito	**RF3.1**		
Nombre	**Inscripción de un familiar en una actividad**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad "Gestión de Inscripciones" será necesario realizar la inscripción de un familiar en una actividad para un año concreto.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la actividad y el familiar deberán existir en el sistema.		

Tabla 15 RF1.1 – Alta de una Actividad

Número de Requisito	**RF3.2**		
Nombre	**Modificación de una Inscripción**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad "Gestión de Inscripciones" será necesario poder modificar una inscripción existente.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la inscripción a modificar debe existir.		

Tabla 16 RF3.2 – Modificación de una Inscripción

Número de Requisito	**RF3.3**		
Nombre	**Borrado de una Inscripción**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad "Gestión de Inscripciones" se dará la posibilidad al usuario de poder eliminar una inscripción previamente realizada.		
Precondiciones	El usuario deberá estar identificado en la aplicación y la inscripción a eliminar existir.		

Tabla 17 RF3.3 – Consulta de una Actividad

Número de Requisito	**RF3.4**		
Nombre	**Renovación de inscripciones**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Dentro de la funcionalidad "Gestión de Inscripciones" se dará la posibilidad al usuario de poder inscribir a un familiar en las mismas inscripciones que ya tiene registradas en la aplicación pero con un año posterior.		
Precondiciones	El usuario deberá estar identificado en la aplicación. El familiar a renovar deberá tener inscripciones registradas y las actividades de dichas inscripciones deberán contar con un registro del año siguiente.		

Tabla 18 RF3.4 – Renovación de Inscripciones

Número de Requisito	**RF4**		
Nombre	**Capacidad para generar listados**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	La aplicación será capaz de generar informes sobre la información almacenada para las actividades, familiares, inscripciones, etc.		
Precondiciones	El usuario deberá estar identificado en el sistema.		

Tabla 19 RF4.1 – Capacidad para generar listados

Número de Requisito	**RF4.1**		
Nombre	**Capacidad exportar los listado a otros formatos**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☐ Alta / Esencial	☒ Media / Deseado	☐ Baja / Opcional
Descripción	La aplicación será capaz de emitir los informes en varios formatos como PDF o Excel.		
Precondiciones	El usuario deberá estar identificado en la aplicación.		

Tabla 20 RF3.3 – Consulta de una Actividad

Número de Requisito	**RF5**		
Nombre	**Capacidad para enviar circulares de correo**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☐ Alta / Esencial	☒ Media / Deseado	☐ Baja / Opcional
Descripción	La aplicación será capaz de enviar correos electrónicos a un listado de familiares que seleccione el usuario de la aplicación.		
Precondiciones	El usuario deberá estar identificado en la aplicación.		

Tabla 21 RF5 – Capacidad para generar listados

Número de Requisito	**RF6**		
Nombre	**Identificación para Acceder al Sistema**		
Tipo	☒ Requisito	☐ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Será necesario identificarse mediante un nombre de usuario y una contraseña de acceso para poder acceder a las funciones de la aplicación.		

Tabla 22 RF5 – Capacidad para generar listados

2.3.1.4 Requisitos no funcionales

Número de Requisito	**RNF1**		
Nombre	**Gestión de Inscripciones por cola de espera**		
Tipo	☐ Requisito	☒ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	Las inscripciones serán gestionadas por un sistema de cola de espera, donde el primero en llegar será el primero en el orden de inscripción. Si en algún momento una inscripción es borrada, todas las inscripciones inmediatamente anteriores avanzarán una posición en la cola.		

Tabla 23 RNF1 – Gestión de Inscripciones por cola de espera

Número de Requisito	**RNF2**		
Nombre	**Sistema 24x7x365**		
Tipo	☐ Requisito	☒ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	La aplicación registra su pico de uso durante las horas del día (de 8h a 18h) pero es muy posible que acabe siendo publicada en Internet y por tanto debe ser capaz de funcionar correctamente y estar disponible en todo momento.		

Tabla 24 RNF2 – Sistema 24x7x365

Número de Requisito	**RNF3**		
Nombre	**Desarrollo Software con modelo Vista-Controlador**		
Tipo	☐ Requisito	☒ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	El tutor del proyecto ha solicitado el desarrollo de la aplicación siguiendo el modelo vista controlador.		

Tabla 25 RNF3 – Desarrollo Software con modelo Vista-Controlador

Número de Requisito	**RNF4**		
Nombre	**La aplicación debe funcionar en dispositivos móviles**		
Tipo	☐ Requisito	☒ Restricción	
Prioridad	☒ Alta / Esencial	☐ Media / Deseado	☐ Baja / Opcional
Descripción	La aplicación debe ser vista y adaptada a dispositivos móviles de forma que permita una interacción eficiente y óptima para el usuario.		

Tabla 26 RNF4 – Desarrollo Software con modelo Vista-Controlador

2.3.2 Casos de Uso

A partir de los requisitos establecidos y tras un análisis en detalle de los mismos se han establecido los siguientes casos de uso que permiten abarcar todas las funcionalidades de la aplicación.

2.3.2.1 CU-001 - Validar Usuario

2.3.2.1.1 Objetivo

Este caso de uso describe como un usuario se identifica en el sistema para comenzar a trabajar.

2.3.2.1.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.1.3 Flujo principal

Actor	Aplicación
1.- El usuario introduce su “Login” y su código pin de acceso.	2.- El sistema valida el usuario y muestra el menú principal con las opciones disponibles para su perfil de usuario.

Tabla 27 Flujo CU-001 - Validar Usuario

2.3.2.1.4 Flujos excepcionales

Si en el paso 2 la información introducida por el usuario no es correcta, la aplicación volverá a la situación inicial.

2.3.2.1.5 Precondiciones

Es necesario que se introduzca un “Login” y una contraseña de acceso que se encuentre registrada en la aplicación previamente.

2.3.2.1.6 Poscondiciones

La aplicación estará con una sesión iniciada y mostrará todas las posibles acciones a realizar.

2.3.2.1.7 Finalización

La aplicación estará con una sesión iniciada y esperando que se seleccione una acción para continuar.

2.3.2.1.8 Requisitos cubiertos

RF6.

2.3.2.2 CU-002 – Rellenar Formulario de Datos

2.3.2.2.1 Objetivo

Este caso de uso describe como un usuario rellena un formulario de datos en la aplicación y cómo este es validado por la misma para poder continuar.

2.3.2.2.2 Inicio

La aplicación estará con una sesión iniciada y en una de las diversas pantallas de formulario con las que cuenta la aplicación esperando a que se introduzcan los datos requeridos.

2.3.2.2.3 Flujo principal

Actor	Aplicación
1.- El usuario rellena el formulario y pulsa el botón que realiza la acción correspondiente al formulario (ej. Alta de una actividad).	2.- El sistema comprueba que los datos del formulario cumplen todos los requisitos y realizada la acción correspondiente del formulario.

Tabla 28 Flujo CU-002 - Rellenar Formulario de Datos

2.3.2.2.4 Flujos excepcionales

Si en el paso 2 la información introducida por el usuario no es correcta, la aplicación mostrará un error al usuario señalando su error o errores en los datos introducidos y se quedará esperando a que el usuario realice una nueva acción.

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.2.5 Precondiciones

La aplicación debe estar con una sesión iniciada y además se ha debido de seleccionar la opción “Alta Familia”, “Alta Actividad”, “Alta Familiar”, “Modificar Actividad”, “Modificar Familia” o “Modificar Familiar”.

2.3.2.2.6 Poscondiciones

La aplicación contará con una actividad, familia o familiar nuevo o modificado y continuará realizando las operaciones correspondientes.

2.3.2.2.7 Finalización

Este caso de uso va a incluido siempre dentro de otros casos de uso así que la finalización dependerá de esos casos de uso que incluyen a este.

2.3.2.2.8 Requisitos cubiertos

RF1 si es una Actividad siempre, más RF1.1 o RF1.2 si es alta o modificación.

RF2 si es una Familia siempre, más RF2.1 o RF2.2 si es alta o modificación de familia y RF2.4 y RF2.4 si es alta o modificación de un familiar.

RF3 si es una Inscripción siempre, más RF3.1 o RF3.2 si es alta o modificación.

2.3.2.3 CU-003 – Alta de una Actividad

2.3.2.3.1 Objetivo

Este caso de uso describe como un usuario registra una nueva actividad en la base de datos de la aplicación.

2.3.2.3.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.3.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Añadir Actividad”.	3.- La aplicación muestra al usuario la pantalla con el formulario para añadir la actividad.
4.- <Include> Rellenar Formulario de datos	
	5.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 29 Flujo CU-003 - Alta de una Actividad

2.3.2.3.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.3.5 Precondiciones

Ninguna.

2.3.2.3.6 Poscondiciones

La aplicación contará con una actividad nueva y el usuario continuará realizando las operaciones correspondientes.

2.3.2.3.7 Finalización

La aplicación contará con una nueva actividad más y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.3.8 Requisitos cubiertos

RF1 y RF1.1.

2.3.2.4 CU-004 – Modificación de una Actividad

2.3.2.4.1 Objetivo

Este caso de uso describe como un usuario modifica una actividad existente en la base de datos de la aplicación.

2.3.2.4.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.4.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Listado”.	3.- La aplicación muestra al usuario la pantalla con el listado de actividades.
4.- El usuario selecciona en el listado la actividad que desea modificar y a continuación hace clic en el botón “Ver detalle”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la actividad.
6.- El usuario hace clic en el botón “Modificar”.	7.- La aplicación muestra al usuario la pantalla con el formulario con los datos de la actividad seleccionada para que modifique lo que estime oportuno.
8.- <Include> Rellenar Formulario de datos	
	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 30 Flujo CU-004 - Modificación de una Actividad

2.3.2.4.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.4.5 Precondiciones

La actividad que se desea modificar debe de existir en la base de datos de la aplicación.

2.3.2.4.6 Poscondiciones

Ninguna.

2.3.2.4.7 Finalización

La aplicación habrá modificado una actividad y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.4.8 Requisitos cubiertos

RF1 y RF1.2.

2.3.2.5 CU-005 – Búsqueda de una Actividad

2.3.2.5.1 Objetivo

Este caso de uso describe como un usuario busca una actividad existente en la base de datos de la aplicación.

2.3.2.5.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.5.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Listado”.	3.- La aplicación muestra al usuario la pantalla con el listado de actividades.
4.- El usuario hace clic en el icono con forma de una lupa para comenzar la búsqueda.	5.- La aplicación muestra al usuario un pop-up con el formulario para que introduzca los criterios de búsqueda.
6.- El usuario introduce los criterios de búsqueda y hace clic en el botón “Buscar”.	7.- La aplicación oculta el pop up de búsqueda y muestra el listado con los resultados encontrados.

Tabla 31 Flujo CU-005 - Búsqueda de una Actividad

2.3.2.5.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.5.5 Precondiciones

Ninguna.

2.3.2.5.6 Poscondiciones

El sistema se hallará en la pantalla con el listado de actividades con el resultado obtenido a la espera de que el usuario realice alguna acción.

2.3.2.5.7 Finalización

La aplicación no habrá realizado ninguna modificación en la base de datos y estará a la espera de que el usuario realice más acciones.

2.3.2.5.8 Requisitos cubiertos

RF1 y RF1.3.

2.3.2.6 CU-006 – Generación de un nuevo Curso

2.3.2.6.1 Objetivo

Este caso de uso describe como un usuario genera un nuevo curso, es decir, replica todas las actividades existentes en la base de datos modificando su curso a uno superior.

2.3.2.6.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.6.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Otros” y a continuación “Cursos”.	3.- La aplicación muestra al usuario la pantalla con el listado de actividades del curso.
4.- El usuario hace clic en el botón “Crear nuevo curso”.	5.- La aplicación generará un nuevo curso y mostrará la información del mismo en el listado.

Tabla 32 Flujo CU-006 - Generación de un nuevo Curso

2.3.2.6.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.6.5 Precondiciones

Ninguna.

2.3.2.6.6 Poscondiciones

Ninguna.

2.3.2.6.7 Finalización

La aplicación contará con un nuevo curso pudiendo seleccionarlo en la pantalla inicial para trabajar con él.

2.3.2.6.8 Requisitos cubiertos

RF1 y RF1.4.

2.3.2.7 CU-007 – Alta de una Familia

2.3.2.7.1 Objetivo

Este caso de uso describe como un usuario registra una nueva familia en la base de datos de la aplicación.

2.3.2.7.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.7.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Añadir Familia”.	3.- La aplicación muestra al usuario la pantalla con el formulario para añadir la actividad.
4.- <Include> Rellenar Formulario de datos	
	5.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 33 Flujo CU-007 - Alta de una Familia

2.3.2.7.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.7.5 Precondiciones

Ninguna.

2.3.2.7.6 Poscondiciones

La aplicación contará con una familia nueva y el usuario continuará realizando las operaciones correspondientes.

2.3.2.7.7 Finalización

La aplicación contará con una nueva familia más y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.7.8 Requisitos cubiertos

RF2 y RF2.1.

2.3.2.8 CU-008 – Modificación de una Familia

2.3.2.8.1 Objetivo

Este caso de uso describe como un usuario modifica una familia existente en la base de datos de la aplicación.

2.3.2.8.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.8.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de todas las familias.
4.- El usuario selecciona en el listado la familia que desea modificar y a continuación hace clic en el botón “Ver Familia”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la familia.
6.- El usuario hace clic en el botón “Modificar datos titular”.	7.- La aplicación muestra al usuario la pantalla con el formulario con los datos de la familia seleccionada para que modifique lo que estime oportuno.
8.- <Include> Rellenar Formulario de datos	
	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 34 Flujo CU-008 - Modificación de una Familia

2.3.2.8.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.8.5 Precondiciones

La familia que se desea modificar debe de existir en la base de datos de la aplicación.

2.3.2.8.6 Poscondiciones

Ninguna.

2.3.2.8.7 Finalización

La aplicación habrá modificado una actividad y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.8.8 Requisitos cubiertos

RF2 y RF2.2.

2.3.2.9 CU-009 – Búsqueda de una Familia

2.3.2.9.1 Objetivo

Este caso de uso describe como un usuario busca una familia existente en la base de datos de la aplicación.

2.3.2.9.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.9.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario	
2.- El usuario selecciona la opción del menú “Familia” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de familias.
4.- El usuario hace clic en el icono con forma de una lupa para comenzar la búsqueda.	5.- La aplicación muestra al usuario un pop-up con el formulario para que introduzca los criterios de búsqueda.
6.- El usuario introduce los criterios de búsqueda y hace clic en el botón “Buscar”.	7.- La aplicación oculta el pop up de búsqueda y muestra el listado con los resultados encontrados.

Tabla 35 Flujo CU-009 - Búsqueda de una Familia

2.3.2.9.4 Flujos excepcionales

En el paso 2 si el usuario desea realizar un primer filtrado simple, puede entrar por alguna de las opciones disponible en vez de por la opción “Ver Todos”. Ej. “Ver Familias Socias”.

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.9.5 Precondiciones

Ninguna.

2.3.2.9.6 Poscondiciones

El sistema se hallará en la pantalla con el listado de actividades con el resultado obtenido a la espera de que el usuario realice alguna acción.

2.3.2.9.7 Finalización

La aplicación no habrá realizado ninguna modificación en la base de datos y estará a la espera de que el usuario realice más acciones.

2.3.2.9.8 Requisitos cubiertos

RF2 y RF2.3.

2.3.2.10 CU-010 – Alta de un Familiar

2.3.2.10.1 Objetivo

Este caso de uso describe como un usuario registra un nuevo familiar para una familia existente en la base de datos de la aplicación.

2.3.2.10.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.10.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de todas las familias.
4.- El usuario selecciona en el listado la familia que desea modificar y a continuación hace clic en el botón “Ver Familia”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la familia.
6.- El usuario hace clic en el botón “Añadir Familiar”.	7.- La aplicación muestra al usuario la pantalla con el formulario con los datos que debe rellenar para añadir un familiar.
8.- <Include> Rellenar Formulario de datos.	
	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 36 Flujo CU-010 - Alta de un Familiar

2.3.2.10.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.10.5 Precondiciones

La familia debe existir en la base de datos de la aplicación.

2.3.2.10.6 Poscondiciones

La aplicación habrá modificado la familia existente añadiendo un nuevo familiar y estará a la espera de nuevas acciones por parte del usuario.

2.3.2.10.7 Finalización

La aplicación habrá modificado la familia existente añadiendo un nuevo familiar y estará a la espera de nuevas acciones por parte del usuario.

2.3.2.10.8 Requisitos cubiertos

RF2 y RF2.4.

2.3.2.11 CU-011 – Modificación de un Familiar

2.3.2.11.1 Objetivo

Este caso de uso describe como un usuario modifica un familiar de una familia existente en la base de datos de la aplicación.

2.3.2.11.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.11.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de todas las familias.
4.- El usuario selecciona en el listado la familia que desea modificar y a continuación hace clic en el botón “Ver Familia”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la familia.
6.- El usuario elegirá del listado de familiares que le muestra la aplicación aquel que quiera modificar y pulsará el botón “Modificar”.	7.- La aplicación muestra al usuario un pop-up con el formulario con los datos del familiar seleccionado para que modifique lo que estime oportuno.
8.- <Include> Rellenar Formulario de datos.	
	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 37 Flujo CU-011 - Modificación de un Familiar

2.3.2.11.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.11.5 Precondiciones

La familia que se desea modificar debe de existir en la base de datos de la aplicación.

2.3.2.11.6 Poscondiciones

La página con la información de la familia recargará el listado de familiares mostrando los cambios al usuario.

2.3.2.11.7 Finalización

La aplicación habrá modificado un familiar y por ende una familia y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.11.8 Requisitos cubiertos

RF2 y RF2.5.

2.3.2.12 CU-012 – Eliminación de un Familiar

2.3.2.12.1 Objetivo

Este caso de uso describe como un usuario modifica una familia existente eliminando a uno de sus familiares de la base de datos de la aplicación.

2.3.2.12.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.12.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de todas las familias.
4.- El usuario selecciona en el listado la familia que desea modificar y a continuación hace clic en el botón “Ver Familia”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la familia.
6.- El usuario elegirá del listado de familiares aquel que quiera eliminar y pulsará el botón “Eliminar”.	7.- La aplicación muestra al usuario un pop-up de confirmación para que confirme la eliminación.
8.- El usuario confirma la operación de borrado.	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 38 Flujo CU-012 – Eliminación de un Familiar

2.3.2.12.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

En el paso 7 la aplicación negará la eliminación del familiar si este tiene inscripciones.

2.3.2.12.5 Precondiciones

La familia que se desea modificar debe de existir en la base de datos de la aplicación.

El familiar a eliminar no puede ser el titular de la familia.

2.3.2.12.6 Poscondiciones

La página con la información de la familia recargará el listado de familiares mostrando los cambios al usuario.

2.3.2.12.7 Finalización

La aplicación habrá eliminado a un familiar modificando con ello a una familia y estará esperando a la realización de más acciones por parte del usuario.

2.3.2.12.8 Requisitos cubiertos

RF2 y RF2.6.

2.3.2.13 CU-013 – Realizar una Inscripción

2.3.2.13.1 Objetivo

Este caso de uso describe como un usuario realiza una inscripción de un miembro de una familia en una actividad dentro de un año concreto.

2.3.2.13.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.13.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Inscripciones”.	3.- La aplicación muestra al usuario la pantalla con el formulario para realizar la inscripción.
4.- El usuario introduce un nombre para buscarlo en la base de datos.	5.- La aplicación muestra al usuario los resultados encontrados para ese nombre.
6.- El usuario elegirá una persona de la lista mostrada y a continuación elegirá una o varias actividades en la que quiere inscribir a esa persona. Finalmente pulsa el botón “Finalizar”.	7.- La aplicación muestra el resultado de la acción al usuario y vuelve al paso número 3 por si el usuario desea continuar.

Tabla 39 Flujo CU-013 – Realizar una inscripción

2.3.2.13.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

En el paso número 3 la aplicación puede que no devuelva resultando obligando al usuario a realizar a empezar en el paso 1 de nuevo.

2.3.2.13.5 Precondiciones

Si se quiere completar el caso de uso al completo deben existir familiares y actividades.

2.3.2.13.6 Poscondiciones

La aplicación contará ahora en la base de datos con una nueva inscripción.

2.3.2.13.7 Finalización

La aplicación se encontrará en un estado como el paso número 3 a la espera de nuevas acciones.

2.3.2.13.8 Requisitos cubiertos

RF3 y RF3.1.

2.3.2.14 CU-014 – Modificar una Inscripción

2.3.2.14.1 Objetivo

Este caso de uso describe como un usuario modifica una inscripción existente para un miembro de una familia en una actividad dentro de un año concreto.

2.3.2.14.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.14.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Listado”.	3.- La aplicación muestra al usuario la pantalla con el listado de actividades.
4.- El usuario selecciona en el listado la actividad que desea modificar y a continuación hace clic en el botón “Ver detalle”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la actividad con tres tablas con los familiares, admitidos, en espera y las bajas.
6.- El usuario elegirá una persona de la tabla de “admitidos” o de la de “en espera” y pulsará el botón modificar.	7.- La aplicación muestra el resultado de la acción al usuario y actualiza las tablas.

Tabla 40 Flujo CU-014 – Modificar una inscripción

2.3.2.14.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.14.5 Precondiciones

Es necesario que haya inscripciones para poder realizar modificaciones.

2.3.2.14.6 Poscondiciones

La aplicación contará ahora en la base de datos con la inscripción seleccionada modificada.

2.3.2.14.7 Finalización

La aplicación se encontrará esperando nuevas acciones por parte del usuario.

2.3.2.14.8 Requisitos cubiertos

RF3 y RF3.2.

2.3.2.15 CU-015 – Eliminar una Inscripción

2.3.2.15.1 Objetivo

Este caso de uso describe como un usuario modifica una inscripción existente para un miembro de una familia en una actividad dentro de un año concreto.

2.3.2.15.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.15.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Actividades” y a continuación “Listado”.	3.- La aplicación muestra al usuario la pantalla con el listado de actividades.
4.- El usuario selecciona en el listado la actividad que desea modificar y a continuación hace clic en el botón “Ver detalle”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la actividad con tres tablas con los familiares, admitidos, en espera y las bajas.
6.- El usuario elegirá una persona de la tabla de “admitidos” o de la de “en espera” y pulsará el botón “Dar de baja”.	7.- La aplicación muestra el resultado de la acción al usuario y actualiza las tablas.

Tabla 41 Flujo CU-015 – Eliminar una inscripción

2.3.2.15.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.15.5 Precondiciones

Es necesario que haya inscripciones para poder realizar bajas.

2.3.2.15.6 Poscondiciones

La aplicación contará ahora en la base de datos con una inscripción menos en la actividad seleccionada.

2.3.2.15.7 Finalización

La aplicación se encontrará esperando nuevas acciones por parte del usuario.

2.3.2.15.8 Requisitos cubiertos

RF3 y RF3.3.

2.3.2.16 CU-016 – Renovación de una Familia

2.3.2.16.1 Objetivo

Este caso de uso describe como un usuario realiza la renovación de una familia, es decir, realizar la inscripción en el año siguiente de los familiares con las actividades del presente año.

2.3.2.16.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.16.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción del menú “Familias” y a continuación “Ver Todos”.	3.- La aplicación muestra al usuario la pantalla con el listado de todas las familias.
4.- El usuario selecciona en el listado la familia que desea modificar y a continuación hace clic en el botón “Ver Familia”.	5.- La aplicación muestra al usuario la pantalla con el detalle de la familia.
6.- El usuario hace clic en el botón “Renovar”.	7.- La aplicación muestra al usuario el formulario para que seleccione a renovar para cada miembro.
8.- <Include> Rellenar Formulario de datos.	
	9.- La aplicación muestra el resultado de la acción al usuario y se queda a la espera de nuevas acciones.

Tabla 42 Flujo CU-016 – Renovación de una Familia

2.3.2.16.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

En el paso 7 la aplicación no dejará renovar actividades donde el familiar no cumpla los requisitos necesarios en el año siguiente (Ej. Es más mayor que la edad máxima de la actividad).

2.3.2.16.5 Precondiciones

Es necesario que haya inscripciones para poder realizar bajas.

Debe existir el año siguiente en la base de datos para poder renovar.

2.3.2.16.6 Poscondiciones

La aplicación contará ahora en la base de datos con inscripciones nuevas para la actividad.

2.3.2.16.7 Finalización

La aplicación se encontrará esperando nuevas acciones por parte del usuario.

2.3.2.16.8 Requisitos cubiertos

RF3 y RF3.4.

2.3.2.17 CU-017 – Generar Listados Tablas

2.3.2.17.1 Objetivo

Este caso de uso describe como un usuario puede realizar listados de cualquiera de las tablas de la aplicación.

2.3.2.17.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.17.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- <Include> Búsqueda Actividad o Familia.	
3.- El usuario hace clic en el botón “XLS”.	4.- La aplicación genera el listado automáticamente y lo descarga en el dispositivo del usuario.

Tabla 43 Flujo CU-017 – Generar Listados Tablas

2.3.2.17.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.17.5 Precondiciones

Ninguna.

2.3.2.17.6 Poscondiciones

La aplicación se quedará a la espera de realizar más acciones por parte del usuario.

2.3.2.17.7 Finalización

La aplicación se quedará a la espera de realizar más acciones por parte del usuario.

2.3.2.17.8 Requisitos cubiertos

RF4 y RF4.1.

2.3.2.18 CU-018 – Generar Correos

2.3.2.18.1 Objetivo

Este caso de uso describe como un usuario puede realizar el envío de un mail genérico para una lista de familiares de la aplicación.

2.3.2.18.2 Inicio

La aplicación estará en la pantalla inicial esperando a que sean introducidos un “Login” y una contraseña de acceso.

2.3.2.18.3 Flujo principal

Actor	Aplicación
1.- <Include> Validar Usuario.	
2.- El usuario selecciona la opción “Otros” y a continuación la opción “Generar Correo” de la barra de menú.	3.- La aplicación muestra al usuario la pantalla con el formulario para el envío del correo.
8.- <Include> Rellenar Formulario de datos.	
4.- El usuario selecciona los destinatarios de entre las opciones posibles y hace “Clic” en enviar.	5.- La aplicación genera el correo y lo envía automáticamente notificándoselo al usuario.

Tabla 44 Flujo CU-018 – Generar Correos

2.3.2.18.4 Flujos excepcionales

El usuario podrá cancelar en cualquiera de sus pasos pulsando el botón de salir.

2.3.2.18.5 Precondiciones

Ninguna.

2.3.2.18.6 Poscondiciones

La aplicación se quedará a la espera de realizar más acciones por parte del usuario.

2.3.2.18.7 Finalización

La aplicación se quedará a la espera de realizar más acciones por parte del usuario.

2.3.2.18.8 Requisitos cubiertos

RF5.

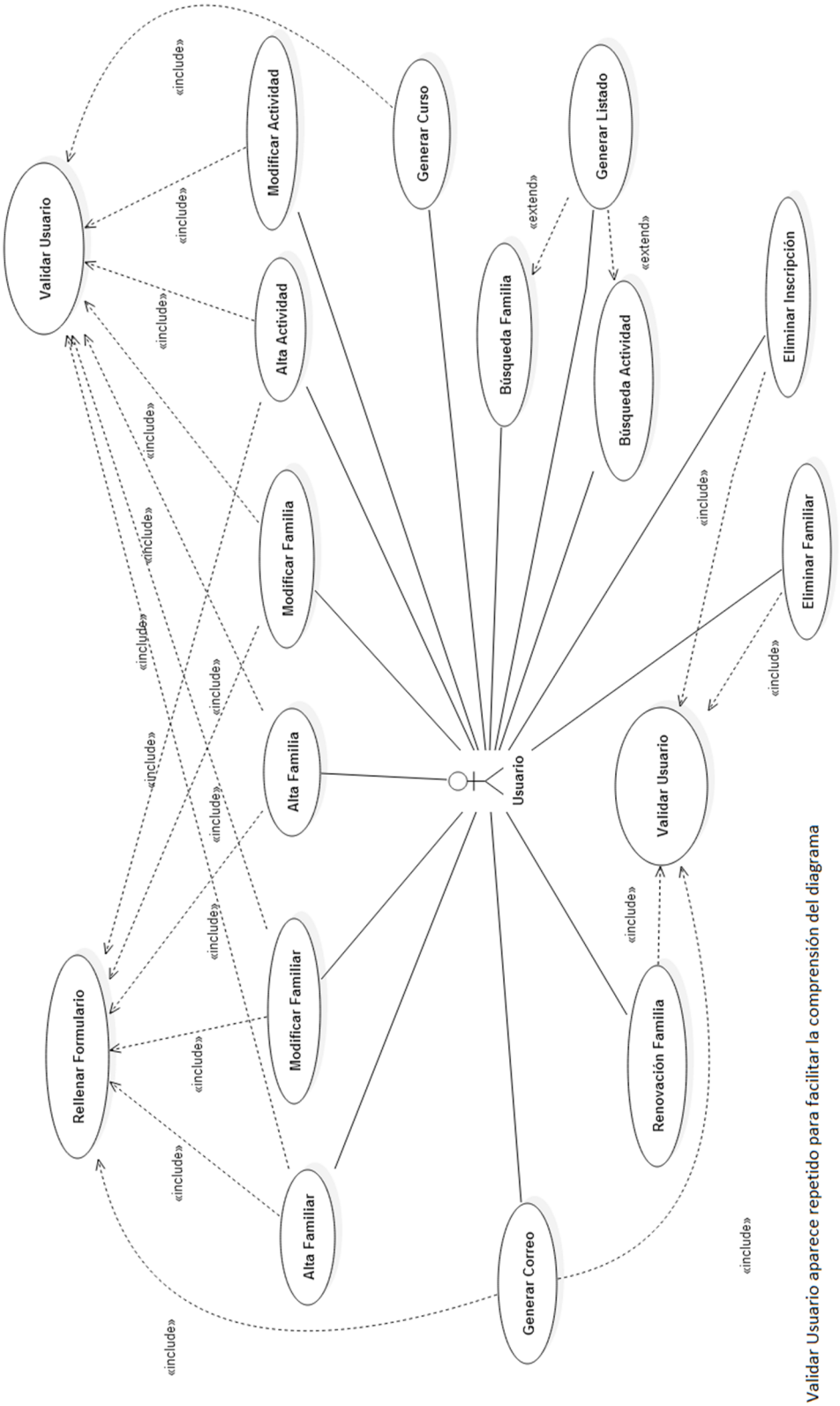

Ilustración 4 Diagrama de Casos de Uso

2.3.3 Diagrama Modelo Entidad-Relación

2.3.3.1 Entidades

Derivado del análisis de requisitos se establecen también las siguientes entidades de datos que serán las encargadas de formar la base necesaria para la implementación de todas las funcionalidades de la aplicación.

- **Familia**

  Utilizada para guardar la información de las familias registradas en el colegio en las que sus miembros participan en actividades.

- **Dato Contacto**

  Como su propio nombre indica, guarda la información de contacto de una familia.

- **Persona**

  Almacena la información relevante sobre una persona que es miembro de una familia.

- **Actividad**

  Tiene la información básica de una actividad realizada en el colegio.

- **Clase**

  Entidad débil respecto a Actividad que almacena la información de una actividad para un año concreto.

- **Horario**

  Entidad débil respecto a Clase que guarda información relacionada con la clase de una actividad, qué día se realiza y a qué hora.

2.3.3.2 Diccionario de Datos

A continuación se expone el diccionario de datos de cada entidad, es decir, las características lógicas de los datos que se almacenar en la base de datos de la aplicación.

- **Familia**

Se ha establecido una clave primaria “NumSocio” que identifica de manera unívoca cada uno de los registros de esta entidad. Además es necesario identificar qué familias son socias y cuales no así que se ha introducido un atributo “esSocio” de tipo booleano. También se necesita saber cuándo se registra una familia así que se incluye un campo “FechaInscripcion”, cuál es el último año que ha pagado y para ello se incluye “CursoPagado” y por último, “NumSolicitud” que permite saber el número de solicitud que recibe una familia cuando inscribe a sus miembros en actividades.

- **Dato Contacto**

Su clave primaria será “IdDatoContacto” y como atributos cuenta con “NombreContacto”, “DetallesContacto”, “Telefono” e “Email” que permiten aportar toda la información relevante sobre un dato de contacto.

- **Persona**

Su clave primaria será “IdPersona” y luego cuenta con los siguientes atributos para poder guardar toda la información necesaria de una persona: “Nombre”, “Apellido1 y 2”, “Tipo” y “FechaNacimiento”. Cabe destacar que el atributo “Tipo” sirve para distinguir el tipo de persona ya sea un padre, hijo, madre, titular u otro.

- **Actividad**

Como clave primaria cuenta con un “IdActividad” y además tiene los atributos “Nombre” y “Descripcion”.

- **Clase**

Su clave primaria es “Curso” más el “IdActividad” de Actividad ya que es una entidad débil y luego cuenta con atributos para acotar la información de una clase como su cupo máximo y mínimo de alumnos, la edad mínima y máxima que se permite en la clase, el responsable de gestionar la actividad y un booleano para indicar si es una actividad llevada internamente por el colegio o bien por alguien externo (Externa).

- **Horario**

Está entidad al ser débil hereda su clave primaria de Clase y cuenta con los atributos de “Dia” más “HoraComienzo” y “HoraFin” para identificar el horario de una actividad.

2.3.3.3 Relaciones entre las entidades

Se ha realizado un diagrama entidad-relación que permite ver cómo las entidades descritas anteriormente se relacionan entre sí y sus propiedades.

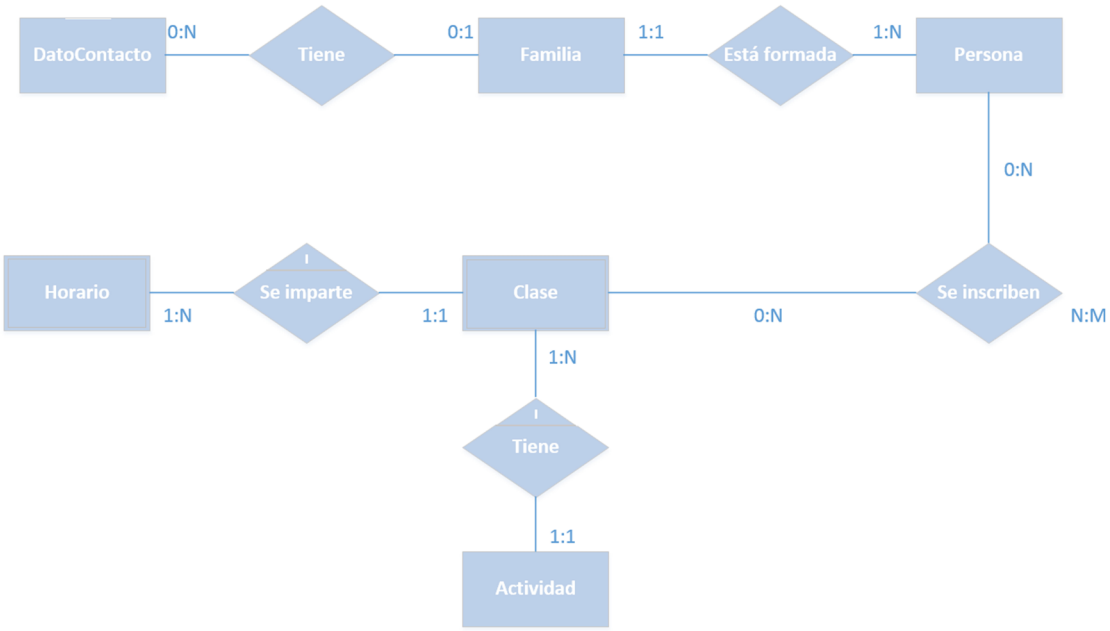

Ilustración 5 Diagrama Entidad-Relación

- **Tiene (Familia – DatoContacto)**

Una familia puede tener ninguno o varios datos de contacto, y un dato de contacto sólo pertenece a una y solo una familia. Relación 1:N.

- **Está Formada (Familia – Persona)**

Una familia puede estar formada por una o varias personas, y una persona sólo pertenece a una y solo una familia. Relación 1:N.

- **Tiene (Actividad – Clase)**

Relación Identificativa siendo Clase una entidad débil. Una actividad puede tener una o varias clases y una clase pertenece a una y sólo a una actividad. Relación 1:N.

- **Se imparte (Clase – Horario)**

Relación Identificativa siendo Horario una entidad débil. Una clase puede tener uno o varios horarios y un horario pertenece a una y sólo a una clase. Relación 1:N.

- **Inscriben (Clase – Persona)**

En una clase se inscriben ninguna o varias personas y una persona puede estar inscrita en ninguna o varias clases. Relación N:M.

2.3.4 Modelo de Datos

Derivado del diseño de entidades de alto nivel que se acaba de exponer se realiza el siguiente diseño modelo relacional de la aplicación.

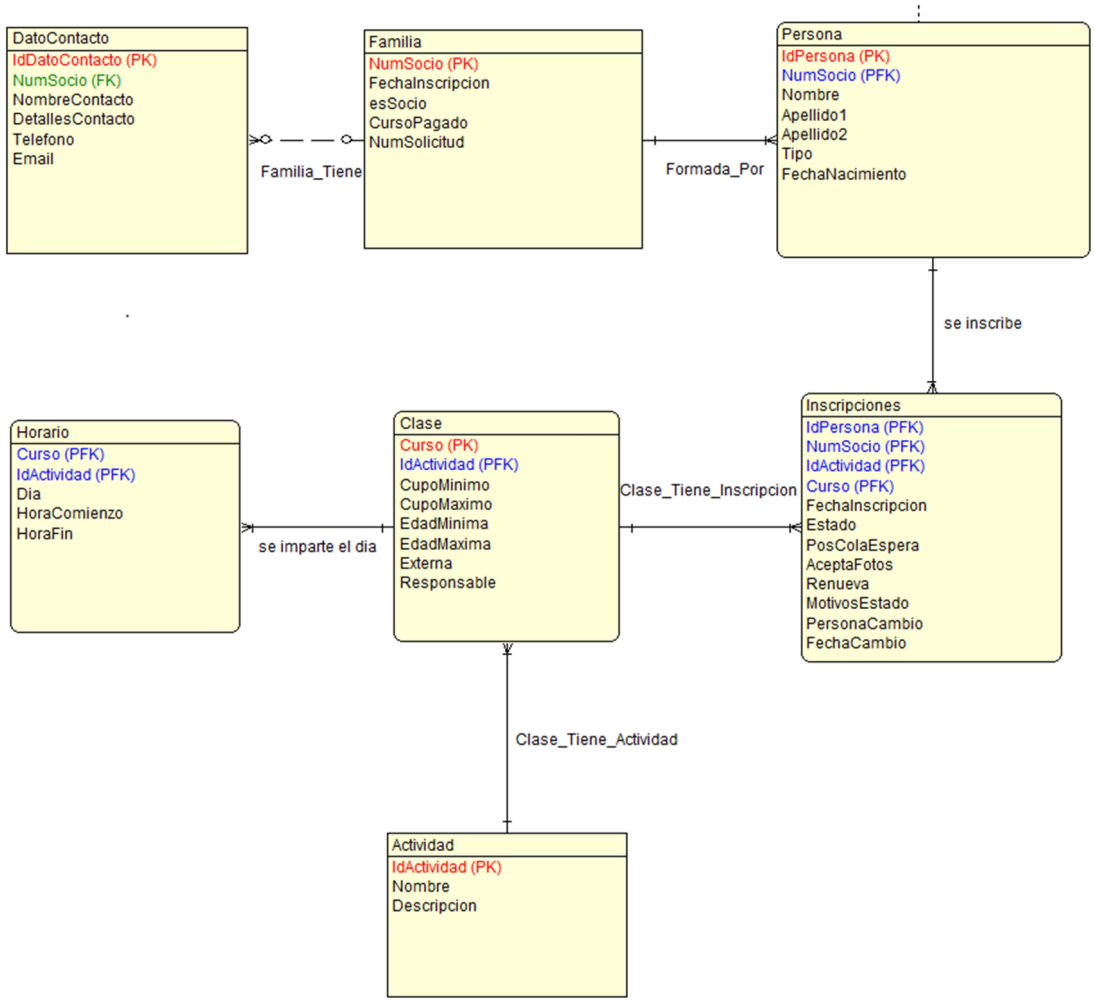

Ilustración 6 Modelo Relacional de la Aplicación

Como se puede apreciar de la relación “Inscriben” de tipo N:M entre las entidades Clase y Persona ha surgido una nueva tabla, “Inscripciones”, que recogerá la información de aquellas personas que se inscriben en clases.

2.3.4.1 Descripción de las Tablas

2.3.4.1.1 Familia

Descripción				
Tabla en la que se guardan los datos de las familias que se registren desde la aplicación.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
NumSocio	Número identificativo de una familia.	INT	Sí	No
FechaInscripcion	Fecha en la que se dio de alta la familia en la aplicación.	DATE	No	No
esSocio	Booleano para saber si una familia es socia o no. 1 = Es Socio. 0 = No es socia.	TINYINT	No	No
CursoPagado	Indica el último año en el que una familia ha pagado la cuota de socio.	INT	No	No
NumSolicitud	Número de Solicitud de la familia obtenido durante el período de inscripción de actividades.	INT	No	No

Tabla 45 Descripción de la tabla Familia

2.3.4.1.2 DatoContacto

Descripción				
Tabla en la que se guarda la información de los contactos que tienen las familias.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
IdDatoContacto	Número identificativo de un dato de contacto para trabajar a nivel interno en la aplicación.	INT	Sí	No
NumSocio	Número identificativo de la familia a la que pertenece el dato de contacto.	INT	No	Sí
NombreContacto	Nombre del contacto.	VARCHAR (120)	No	No
DetallesContacto	Breve explicación sobre el contacto.	VARCHAR (500)	No	No
Telefono	Teléfono del contacto.	INT	No	No
Email	Email del contacto.	VARCHAR (100)	No	No

Tabla 46 Descripción de la tabla DatoContacto

2.3.4.1.3 Persona

Descripción				
Tabla en la que se guarda la información de los miembros que tienen las familias.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
IdPersona	Número identificativo de una persona para trabajar a nivel interno en la aplicación.	INT	Sí	No
NumSocio	Número identificativo de la familia a la que pertenece la persona.	INT	Sí	Sí
Nombre	Nombre de la persona.	VARCHAR (50)	No	No
Apellido1	Primer Apellido de la persona.	VARCHAR (100)	No	No
Apellido2	Segundo Apellido de la persona.	VARCHAR (100)	No	No
Tipo	Para distinguir que tipo de miembro de la familia es. Sus posibles valores son: SOCIO, PADRE, MADRE, HIJO, HIJA, OTRO.	ENUM	No	No
FechaNacimiento	Año de nacimiento de la persona.	INT	No	No

Tabla 47 Descripción de la tabla Persona

2.3.4.1.4 Actividad

Descripción				
Tabla en la que se guarda la información de las actividades disponibles en la aplicación.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
IdActividad	Número identificativo de una actividad para trabajar a nivel interno en la aplicación.	INT	Sí	No
Nombre	Nombre de la actividad.	VARCHAR (100)	No	No
Descripcion	Descripción de la actividad.	VARCHAR (500)	No	No

Tabla 48 Descripción de la tabla Actividad

2.3.4.1.5 Clase

Descripción				
Tabla en la que se guarda la información de las clases que tienen las actividades de la aplicación.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
Curso	Curso al que pertenece la clase.	INT	Sí	No
IdActividad	Número identificativo de la actividad a la que pertenece el registro de la clase.	INT	Sí	Sí
CupoMinimo	Número mínimo de alumnos que necesita la actividad para poderse realizar.	SMALLINT	No	No
CupoMaximo	Número máximo de alumnos que permite tener inscritos la actividad.	SMALLINT	No	No
EdadMinima	Edad mínima que pueden tener las personas que deseen realizar la actividad.	SMALLINT	No	No
EdadMaxima	Edad máxima que pueden tener las personas que deseen realizar la actividad.	SMALLINT	No	No
Externa	Booleano para saber si la clase es gestionada por el colegio o por alguien externo. 1 = Es externa, 0 = no es externa.	TINYINT	No	No
Responsable	Datos del responsable encargado de gestionar la clase desde la aplicación.	VARCHAR (60)	No	No

Tabla 49 Descripción de la tabla Clase

2.3.4.1.6 Horario

Descripción				
Tabla en la que se guarda la información de los horarios de las clases que tienen las actividades de la aplicación.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
Curso	Curso de la clase a la que pertenece el horario.	INT	Sí	Sí
IdActividad	Número identificativo de la actividad a la que pertenece el registro de la clase de este horario.	INT	Sí	Sí
Dia	Día de la semana en el que se celebra la clase de la actividad.	VARCHAR (20)	No	No
HoraComienzo	Hora a la que comienza la clase.	VARCHAR (20)	No	No
HoraFin	Hora a la que finaliza la clase.	VARCHAR (20)	No	No

Tabla 50 Descripción de la tabla Horario

2.3.4.1.7 Inscripciones

Descripción				
Tabla en la que se guarda la información de las inscripciones que se realizan en la aplicación.				
Nombre Columna	**Definición**	**Tipo de Dato**	**PK**	**FK**
IdPersona	Identificativo de la Persona que se inscribe en una clase de una actividad.	INT	Sí	Sí
NumSocio	Número identificativo de la familia de la persona que se inscribe en una clase de una actividad.	INT	Sí	Sí
IdActividad	Número identificativo de la actividad a la que pertenece el registro de la inscripción.	INT	Sí	Sí
Curso	Curso de la clase de la actividad en la que la persona realiza la inscripción.	INT	Sí	Sí
FechaInscripcion	Fecha en la que se realiza la inscripción.	DATETIME	No	No
Estado	Estado en el que se encuentra la inscripción. Es un enumerado que permite los valores: ADMITIDO, BAJA, ESPERA.	ENUM	No	No
PosColaEspera	Posición en la cola de espera de la inscripción. Puede ser nulo si es una solicitud admitida.	SMALLINT	No	No
AceptaFotos	Booleano para saber si la persona inscrita acepta que se le realicen fotos o no. 1 = Acepta fotos, 0 = no las acepta.	TINYINT	No	No
Renueva	Enumerado para indicar si una persona inscrita y admitida en una actividad renueva para esa misma actividad en el año siguiente.	TINYINT	No	No
MotivosEstado	Detalles del porqué del estado de la inscripción.	VARCHAR (500)	No	No
PersonaCambio	Datos de la persona que ha realizado cambios en la inscripción desde la aplicación.	VARCHAR (100)	No	No
FechaInscripcion	Fecha del último cambio que ha sufrido la inscripción desde la aplicación.	DATETIME	No	No

Tabla 51 Descripción de la tabla Inscripciones

2.3.5 Diseño Técnico

Este apartado explica en detalle cómo se ha realizado la arquitectura de toda la aplicación para conseguir desarrollarla prestando un especial enfoque a la aplicación del patrón MVC.

La aplicación se ha desarrollado de forma que el objetivo de MVC, dividir la aplicación en tres capas claramente diferenciadas aislando así el componente de datos de la aplicación (modelo) del estado del sistema del portal de actividades extraescolares (controlador), así como de la representación visual de ésta (vista), dotándole de este modo al proyecto de independencia, encapsulamiento y robustez que favorecen su mantenibilidad y escalabilidad.

En la siguiente imagen podemos ver un diagrama que muestra los principales componentes software que integran la aplicación (navegador, JSPs, Servlets, etc.), donde podemos observar cómo interactúan unos con otros y su pertenencia a cada categoría del patrón MVC.

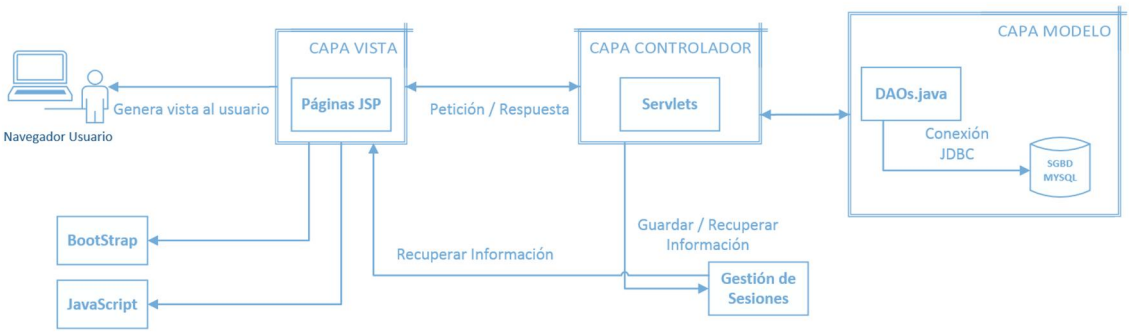

Ilustración 7 Diagrama componentes de la aplicación

2.3.5.1 Capa Modelo

Como se ha explicado anteriormente el modelo en el patrón MVC es la capa encargada de la representación de la información con la cual la aplicación opera, por lo tanto se encarga de gestionar todos los accesos a dicha información.

El primer paso es conseguir el acceso a la base de datos. En este aspecto, se ha implementado una clase sencilla denominada “DBConnection” que haciendo uso de la API JDBC realiza una conexión con la base de datos y permite realizar operaciones contra ella. Gracias a JDBC es posible conectarse a una base de datos independientemente del sistema operativo donde se ejecute o del tipo de sistema gestor de base de datos al cual se accede, ya que la propia API se adapta utilizando el dialecto SQL del modelo de base de datos que se le indique. Esto permite dotar al proyecto de una facilidad inmensa para poderlo portar en algún momento si se desea a cualquier otro sistema gesto de base de datos.

El siguiente paso es conseguir interpretar la información que recuperamos de la base de datos. Para ello se han desarrollado una clase por cada una de las tablas que tiene la base de datos de la aplicación. Estas clases tienen todas como atributos cada uno de los atributos que tiene su tabla homónima en la base de datos encapsulados con métodos getters y setters. Además algunas de ellas se aprovechan una de las características de la programación orientada a objetos como es la herencia permitiendo guarda la relación entre las distintas clases como es en el caso entre Clase y Actividad, donde Clase hereda de Actividad consiguiendo así acceso a sus propiedades y características.

Como JDBC es imposible que utilizando su API conozca las clases que hemos creado resulta necesario establecer unos componentes que suministren una interfaz común entre la aplicación y la base de datos. Estos componentes se denominan Objetos de Acceso a Datos (en adelante DAOs) y permiten aislar a una aplicación de la tecnología de persistencia Java subyacente. Usando DAOs significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación. Es importante señalar que la utilización de DAOs influye en el rendimiento de la aplicación ya que supone utilizar más código y más tiempo de procesamiento por tanto; pero en el caso de este proyecto no es algo relevante ya que la aplicación no está destinada a sufrir unas exigencias de rendimiento elevadas.

Al igual que como con las clases que implementan el modelo de datos, se han creado tantos DAOs como tablas existen en la base de datos, es decir, un total de 7 DAOs. Todos ellos implementan como mínimo funciones para realizar un INSERT, UPDATE y DELETE en la tabla de la que realizan el interfaz; y por lo general todos cuentan con mínimo un método que realiza una función SELECT para rescatar datos de la base de datos.

Sirva de ejemplo DAOHorarios que implementa la interfaz de comunicación con la tabla Horario de la base de datos. Esta clase cuenta con dos métodos para insertar, uno para actualizar, uno para borrar y uno que realizar una consulta para obtener el horario u horarios de una clase concreta.

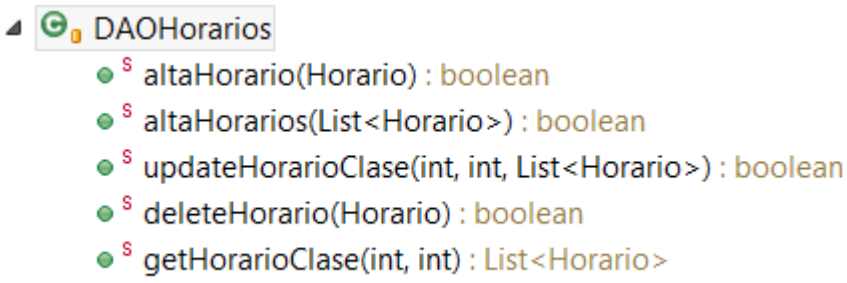

Ilustración 8 Ejemplo de DAO - DAOHorarios

A continuación mostramos un diagrama de clases que muestra el resultado de la integración del modelo de datos en la lógica de la aplicación.

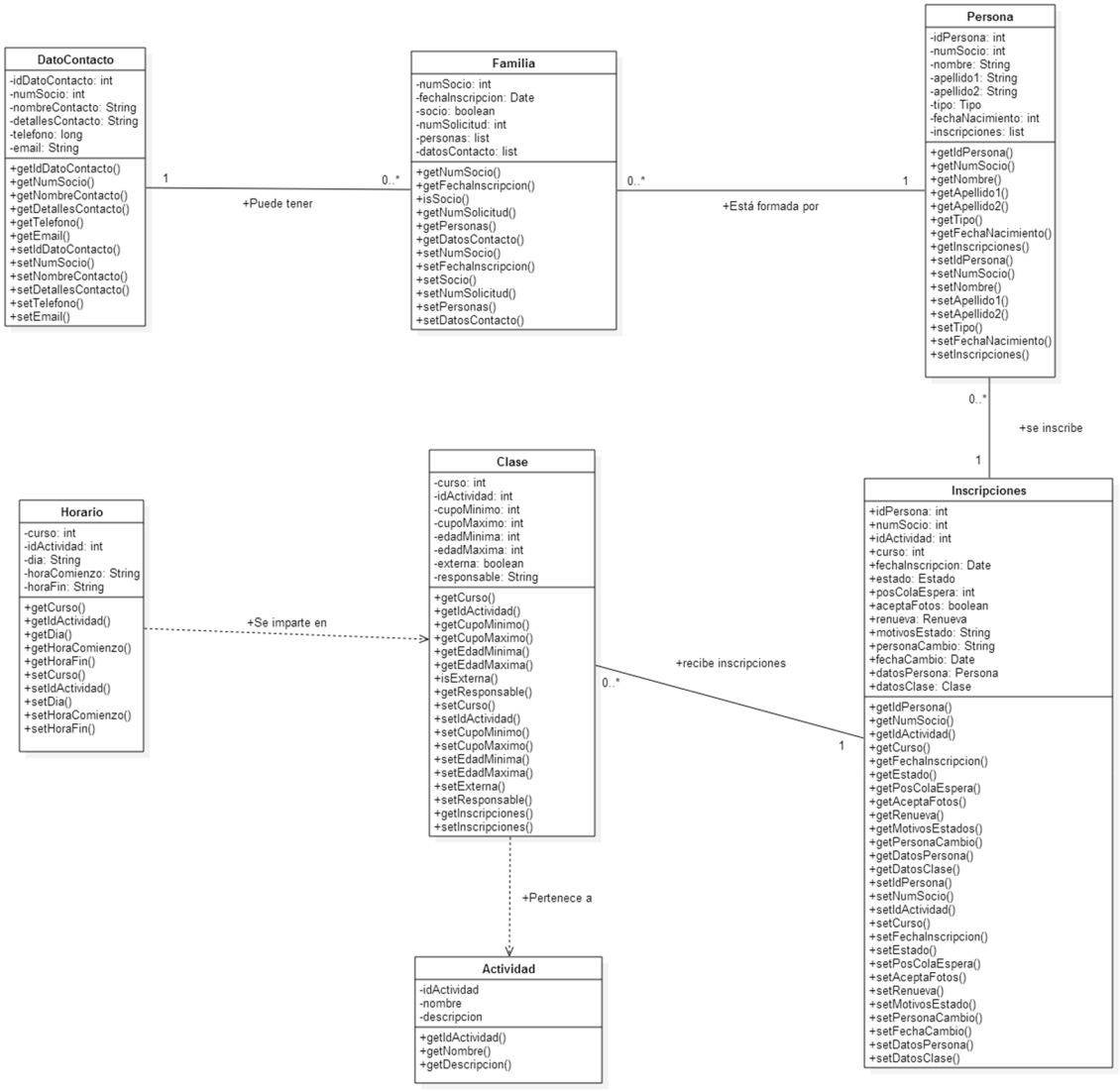

Ilustración 9 Diagrama de Clases

2.3.5.2 Capa Controlador

La capa controlador es la encargada de manejar los eventos provocados por el usuario y comunicarse con el modelo para generar la respuesta / acción al evento y, si es necesario, generar una respuesta al usuario.

En la aplicación la capa controlador es gestionada mediante Servlets de JAVA. Se encargan de recibir las peticiones de las páginas JSP desarrolladas y conectar con los DAOs para el acceso a la base de datos o con alguna de las clases creadas para el envío de emails o de generación de listados y posteriormente distribuir la navegación del usuario a través de la página redireccionando cuando sea necesario o comandando un resultado de servlet que provoque la recarga de un elemento de la página.

Se han implementado tantos servlets como operaciones concretas han sido necesarias y en la aplicación se han distribuido en paquetes en función de lo que soliciten las peticiones. Así se tienen los siguientes paquetes:

- **Alta**

  Aquí se ubican los servlets que comandan operaciones de añadido en la base de datos, INSERT en alguna de las tablas, como son dar de Alta una familia, realizar una inscripción, crear una actividad, etc.

- **Delete**

  Aquí se ubican los servlets que comandan operaciones de borrado en la base de datos, DELETE en alguna de las tablas, como son dar de baja una inscripción, eliminar un curso, etc.

- **Edit**

  Aquí se ubican los servlets que comandan operaciones de edición en la base de datos, UPDATE en alguna de las tablas, como son editar una familia o familiar, etc.

- **Get**

  Aquí se ubican los servlets que comandan operaciones de solicitud de datos a la base de datos, consultas SELECT, como son solicitar el listado de familias, las actividades inscritas de un familiar, etc.

- **Sin clasificar**

  Son servlets que se encargan de operaciones que no tienen cabida en ninguna de las secciones anteriores por ser muy específicas sus acciones como realizar el login de un usuario, enviar email, o establecer un curso con el que trabajar distinto, etc.

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {


            if(request.getParameter("formEditActividadId")!= null
&& request.getParameter("formEditActividadCurso")!= null ){

                  Integer idActividad =
Integer.parseInt(request.getParameter("formEditActividadId"));
                  Integer curso
=Integer.parseInt(request.getParameter("formEditActividadCurso"))
;

                  Clase claseDatos =
DAOClases.getClase(idActividad, curso);
                  request.setAttribute("datosClase", claseDatos);

                  request.setAttribute("horarioClase",
DAOHorarios.getHorarioClase(idActividad,curso));


      request.getRequestDispatcher("/pages/modificarActividad.jsp"
).forward(request, response);

            }


            }
```

Ilustración 10 Ejemplo de Servlet de la aplicación

2.3.5.2.1 Clase Filter para el control de la sesión

La clase Filter dentro del paquete servlet de la especificación J2EE permite capturar las peticiones realizadas por un usuario dentro de una página web, con independencia de que sean peticiones provenientes de JSP o un Servlet, y analizarlas y/o transformarlas.

Gracias a Filter podemos establecer filtros de una manera muy sencilla siendo altamente reutilizables consiguiendo de este modo un gran poder dentro de una aplicación web. Ejemplos de uso comunes son controlar el acceso a la aplicación, transformaciones, auditoria, registro, cifrado, manejo de sesiones, etc.

Un recurso habitual en las aplicaciones web es guardar atributos en sesión para utilizarlos a lo largo de todas las páginas de la aplicación. Esto provoca que si el usuario deja morir su sesión mostrando inactividad durante un tiempo establecido por la aplicación, la sesión se ha destruido y si el usuario solicita una página que maneje datos de la sesión, se producirán errores y resultados no esperados. Implementar un filtro facilitará la tarea de controlar el timeout de la sesión. Esta práctica se ha llevado a cabo en este proyecto para controlar que siempre hay un usuario logado, que exista conexión con la base de datos.

Para ello, en primer lugar se ha creado una clase que implementa la interfaz Filter, “FiltroSesion”. Dicha clase contiene el método doFilter() que verifica la existencia en sesión (HttpSession) de la variable “cursoActual” que guarda el curso con el que está trabajando el usuario y además comprueba que la conexión con la base de datos siga existiendo. En caso de que no estén disponibles, se redirigirá al usuario a la página de log in, para que proceda a la autenticación nuevamente.

Por último es necesario incluir en el archivo de despliegue de la aplicación, el fichero web.xml, sobre que peticiones se desea aplicar el filtro implementado. Para esta aplicación se ha configurado para que pasen por todas las peticiones JSP y todos los servlets.

```
<!-- Creación del filtro -->
  <filter>
    <filter-name>FiltroSesion</filter-name>
    <filter-class>filter.FiltroSesion</filter-class>
  </filter>
  <!-- Mapeo del filtro sobre las peticiones de JSP -->
  <filter-mapping>
    <filter-name>FiltroSesion</filter-name>
    <url-pattern>/pages/*</url-pattern>
  </filter-mapping>
  <!-- Mapeo del filtro sobre las peticiones de Servlets -->
  <filter-mapping>
    <filter-name>FiltroSesion</filter-name>
    <url-pattern>/srv/*</url-pattern>
  </filter-mapping>
```

Ilustración 11 Implementación de Filtrado en web.xml

2.3.5.3 Capa Vista

La capa vista hace una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

En esta aplicación para la vista se han usado páginas JSP apoyadas para su diseño y funcionamiento en Bootstrap, JavaScript (con jQuery y jqGrid).

Como el menú de la aplicación siempre está presente en todas las aplicaciones se ha diseñado como una página aparte y se ha incluido en el resto aprovechando la capacidad de JSP de incluir ficheros JSP dentro de otros ficheros JSP. De esta manera modularizamos el diseño de las páginas y podemos reutilizarlo, mantenerlo de una manera sencilla.

También se han diseñado las páginas intentando conseguir que sean lo más usable posible manteniendo un diseño lo más uniforme posible a lo largo de todas las páginas. Un ejemplo de ello es el posicionamiento de los botones a la hora de rellenar formularios colocando las opciones de cancelar/volver a la izquierda y la de guardar a la derecha ya que en la cultura occidental leemos de izquierda a derecha y tendemos a hacerlo en diagonal lo que provoca que el orden natural de una acción probable en pantalla va a estar situada a la derecha y no a la izquierda.

2.3.6 Conclusiones

La principal conclusión extraída de este proyecto es que se ha logrado establecer el objetivo por el que fue planteado ya que se ha conseguido una aplicación capaz de administrar las actividades extraescolares de un entorno docente, en este caso un colegio.

También destacamos el hecho de que actualmente resulta excesivamente sencillo realizar una aplicación web mediante el uso del patrón MVC y tecnologías como Bootstrap o JSP. Esto hace plantearse el debate de si las aplicaciones clásicas de escritorio han pasado a mejor vida ya que con tanta facilidad resulta bastante coherente apoyar un desarrollo de aplicativo web aunque este finalmente no vaya a salir a red exterior en un principio ya que el futuro es incierto y quién sabe si más adelante no será necesario salir a Internet y entonces ya tendremos más de la mitad del trabajo desarrollado.

Destacable también la facilidad para crear un diseño atractivo a la par que útil en diferentes dispositivos de Bootstrap ya que para el comienzo de este proyecto se desconocía por completo cómo trabajar con este framework y el resultado ha sido bastante positivo.

Por último, destacar que al margen de haber conseguido el objetivo del proyecto se ha conseguido establecer una base que puede ser continuada o adaptada fácilmente ya que el código ha sido desarrollado aplicando las técnicas de la programación orientada a objetos de herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

2.3.7 Trabajo a futuro

Es cierto que se ha conseguido el objetivo del proyecto pero también es cierto que no hay aplicación que no necesite mantenimiento y modificaciones con el paso del tiempo de su ciclo de vida.

A las habituales tareas de mantenimiento, corrección de bugs y adición de nuevas necesidades es recomendable señalar también que habría que incluir:

- Añadir la posibilidad de tener distintos perfiles de usuarios con distintas funcionalidades y responsabilidades sobre la aplicación.
- Aplicar un sistema de cifrado de la información para asegurar la confidencialidad en todo momento de los datos.
- Implementar un sistema de backup & restoration de la base de datos que puedan lanzar los usuarios de la aplicación.
- Inclusión de un sistema de pruebas de estrés y rendimiento.

Capítulo 3
3 Pliego de Condiciones

En este apartado se expresan las condiciones necesarias para poder utilizar todo lo expuesto en este proyecto.

3.1 Condiciones Generales

Como ya se ha indicado, para el desarrollo del proyecto se ha utilizado diverso software, en la mayoría de casos libre, y en otros con versiones orientadas para entornos domésticos o docentes:

- MySQL cuenta con una licencia GNU (General Public License) en su versión 2.
- Bootstrap se publica bajo la licencia "Apache 2 License" y está protegido por el siguiente copyright: "copyright 2013 Twitter".
- JavaScript cuenta con una licencia GNU General Public License en su versión 3.
- JQuery está licenciado bajo MIT.
- JqGrid está licenciado bajo la licencia Creative Commons v3.
- JSP cuenta con una licencia GNU en su versión 2.
- JAVA se utiliza bajo un acuerdo de licencia Oracle Binary Code License Agreement.
- Apache Tomcat se publica bajo licencia Apache License version 2.

3.2 Condiciones de Materiales y Equipos

En cuanto a condiciones materiales, todo el trabajo se desarrolla con un ordenador de sobremesa aunque lo bueno de ser una aplicación web realizada en JAVA y MySQL es que permite su desarrollo bajo cualquier plataforma sin necesitar nada más que una Java Virtual Machine (JVM en adelante), el servidor Apache Tomcat y MySQL para alojar la aplicación y la base de datos. Además tampoco es necesario un equipo muy potente aunque si se recomienda porque a mayor potencia, mejor rendimiento de la base de datos MySQL. A continuación mostramos las características del ordenador utilizado.

Procesador Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz, 3501 Mhz, 4 procesadores principales, 4 procesadores lógicos.

Memoria RAM 8.00 GB.

Sistema Operativo Windows 8.1 Pro.

Software Instalado

- Apache Tomcat 7.0.

- Java Development KIT 8 Update 45.
- MySQL 5.5.44.
- Eclipse Java EE IDE for Web Developers.Version: Luna Service Release 1 (4.4.1) Build id: 20140925-1800.

3.3 Condiciones de Ejecución

Al igual que para el desarrollo, para ejecutar la aplicación basta con tener una JVM y el servidor Apache Tomcat junto el SGBD MySQL con la base de datos instalados.

Capítulo 4

4 Configuración entorno de desarrollo

Esta sección explica todo los componentes software a instalar en un ordenador para poder desplegar el proyecto completo y ver o modificar su código fuente.

Para la instalación se utilizarán los ficheros proporcionados en el CD incluido junto la documentación de este proyecto. En concreto hay que realizar la instalación de cuatro componentes software, el JDK de JAVA, Apache Tomcat 7, Eclipse IDE, MySQL 5, y la importación del proyecto dentro del IDE Eclipse.

Es posible que para alguno de los componentes se redirija a la sección “Manual de Instalación” ya que se ha agrupado todo allí para evitar redundancia y alargar sin necesidad la presente memoria.

Todo el entorno se explica para un ordenador con sistema operativo Windows debido a su presencia mayoritaria en la mayoría de los hogares pero se recuerda que el proyecto es multiplataforma y podría realizarse la instalación de los mismos componentes en otros sistemas operativos. Además en el CD de instalación se incluyen las versiones de 32 bits y de 64 bits para aquellos programas que cuentan con distintos ficheros de instalación.

4.1 Instalación del JDK 8 de JAVA

Seguir pasos indicados en el apartado 5.1 Instalación del JDK 8 de JAVA.

4.2 Instalación de Apache Tomcat 8

Seguir pasos indicados en el apartado 5.2 Instalación de Apache Tomcat 8.

4.3 Instalación de MySQL

Seguir pasos indicados en el apartado 5.3 Instalación de MySQL.

4.4 Instalación de Eclipse IDE

Para realizar esta instalación se utilizará el fichero “eclipse-jee-luna-SR1-win32-x86_64” y bastará con descomprimirlo en el directorio donde deseemos instalarlo. Se recomienda usar una ruta lo más corta posible y sin espacios como por ejemplo “C:\” ya que la estructura de directorios de Eclipse es muy larga y si la ruta de instalación es muy larga puede presentar problemas a la hora de descomprimir los ficheros.

4.4.1 Añadir servidor Apache Tomcat a Eclipse

Para añadir un servido a Eclipse se escogerá la opción del menú “File -> New -> Others”. En la ventana que aparecerá a continuación, se escogerá la opción “Server” dentro de la carpeta con el mismo nombre.

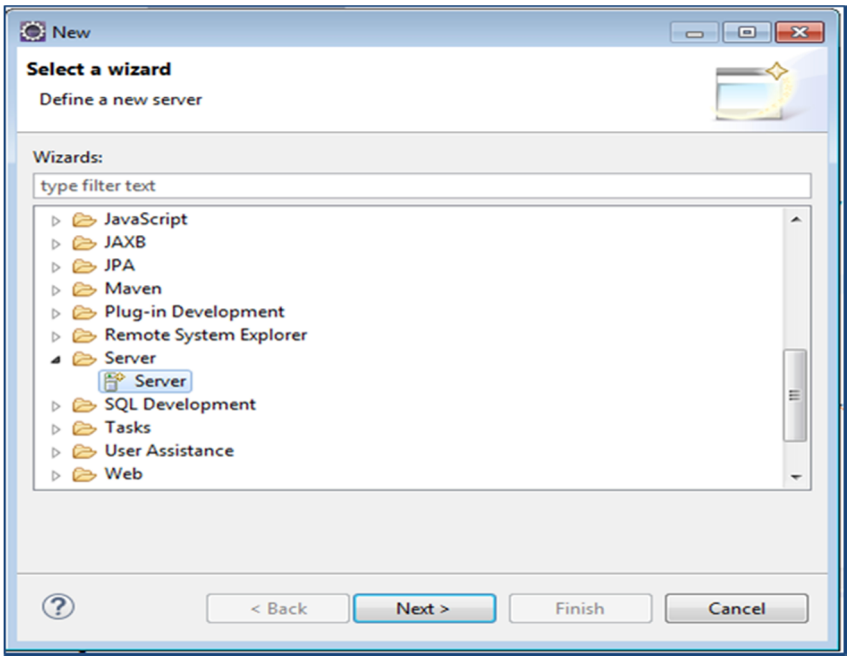

Ilustración 12 Instalación de Eclipse - Añadir Servidor 1

En la siguiente ventana, dentro de la carpeta Apache, seleccionamos la versión instalada en el equipo; para nuestro caso es la versión 7.0. Después se pulsa el botón Next.

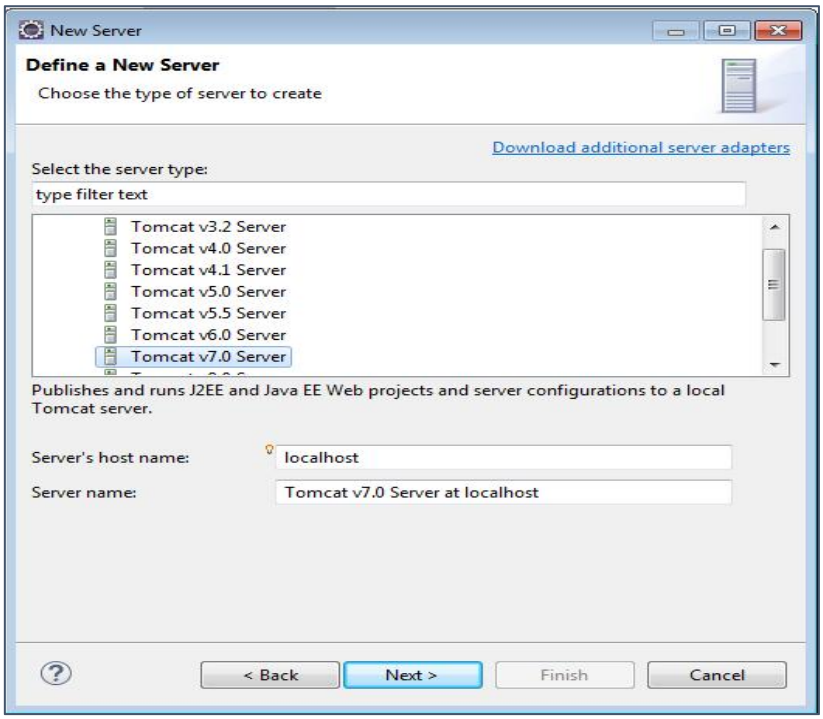

Ilustración 13 Instalación de Eclipse - Añadir Servidor 2

A continuación es necesario indicar el directorio donde tenemos instalado Apache Tomcat y el JRE de la versión Java a utilizar que en este caso el 7. Para terminar pulsamos Finish.

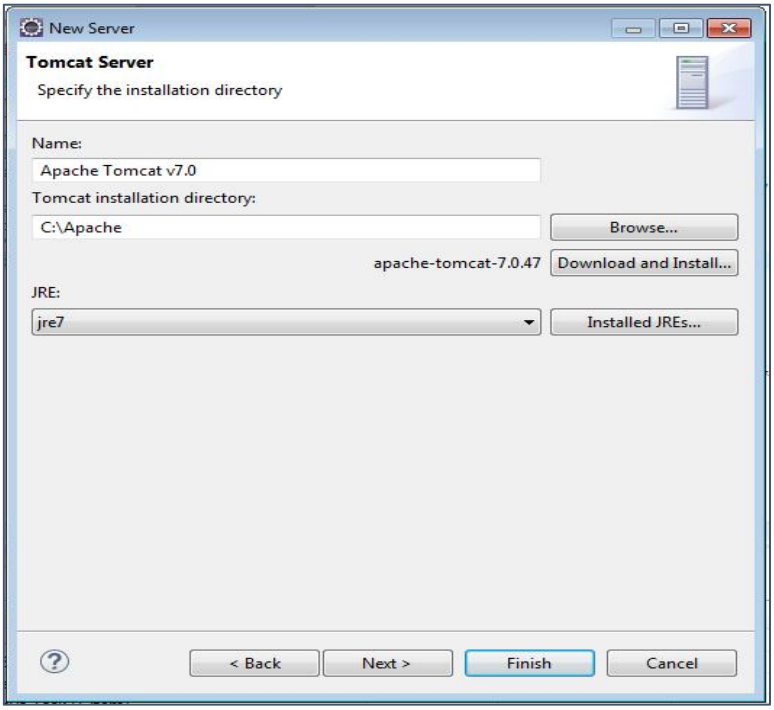

Ilustración 14 Instalación de Eclipse - Añadir Servidor 3

Una vez finalizado el proceso, en el explorador de proyectos de Eclipse, deberá haberse creado una carpeta Server cuyo contenido será el servidor que acabamos de añadir.

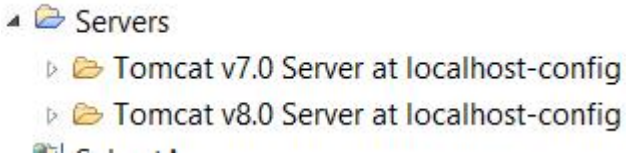

Ilustración 15 Instalación de Eclipse - Añadir Servidor 4

4.4.2 Importar Proyecto a Eclipse

Para poder ver el código de la aplicación es necesario exportar el proyecto al IDE Eclipse para poder tener acceso a su estructura de ficheros. Para ello se abrirá Eclipse y se accederá en el menú a “File” y se seleccionará la opción “Import”.

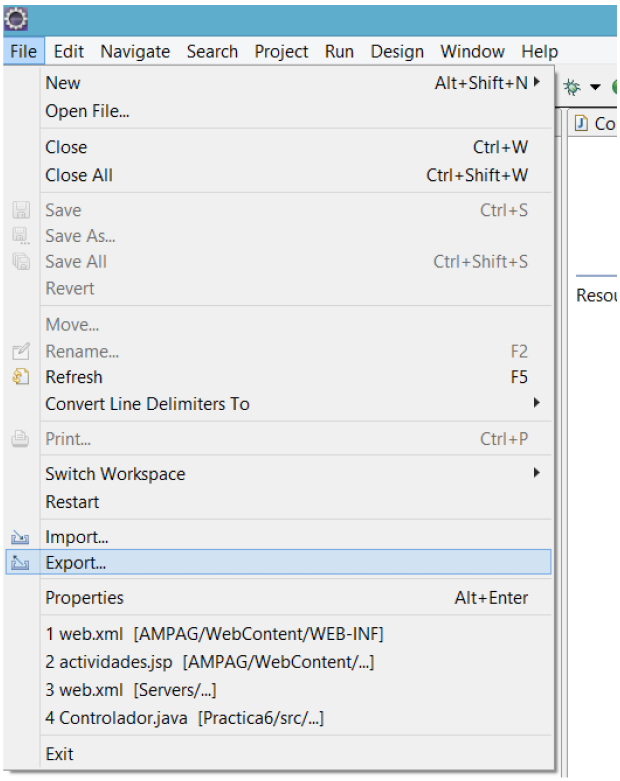

Ilustración 16 Eclipse - Importar Proyecto 1

Eclipse mostrará una pantalla y se elegirá la opción “Existing Projects into Workspace” y se pulsará el botón “Next”.

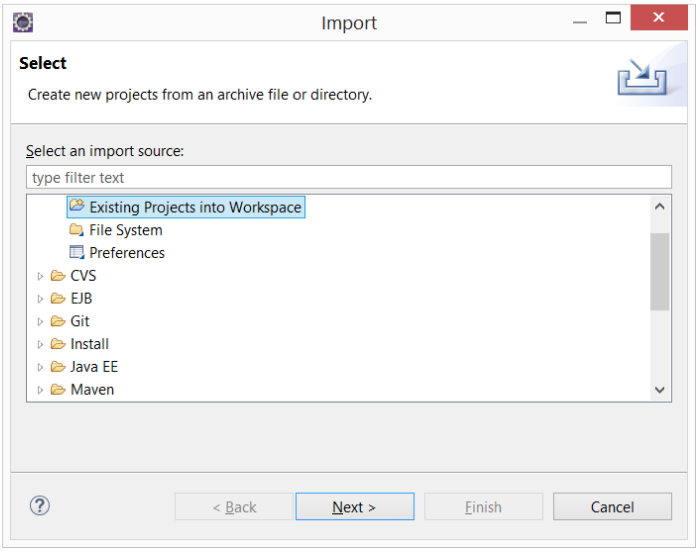

Ilustración 17 Eclipse - Importar Proyecto 2

En la siguiente pantalla se seleccionará en “Root Directory” la ruta del cd de instalación con la carpeta “AMPAG”. Será necesario asegurarse que en el recuadro “Projects” está seleccionado “AMPAG” y que el checkbox “Copy Projects into Workspace” se encuentra también marcado. Finalizaremos pulsando el botón “Finish”.

Una vez haya terminado Eclipse de importar el proyecto, este aparecerá en la parte izquierda de la pantalla con todo su árbol de directorios y estará todo preparado para comenzar a trabajar con el proyecto si así lo deseamos.

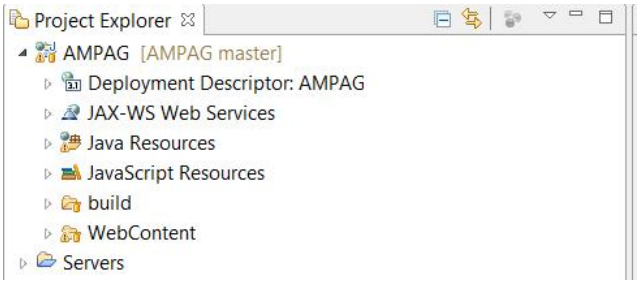

Ilustración 18 Eclipse - Importar Proyecto 3

4.4.3 Crear fichero WAR para desplegar la aplicación

Una vez se haya modificado la aplicación y se desee generar una versión para publicarla en Apache Tomcat será necesario realizar un fichero WAR que es el fichero que contendrá todo lo necesario para que Apache Tomcat despliegue la aplicación. A continuación se muestran los pasos a realizar para conseguir generar un fichero de este tipo en Eclipse.

El primer paso será acceder al menú del proyecto haciendo doble clic en su nombre y a continuación iremos a la opción “Export” y de las opciones que se nos muestran acto seguido escogeremos “WAR File”.

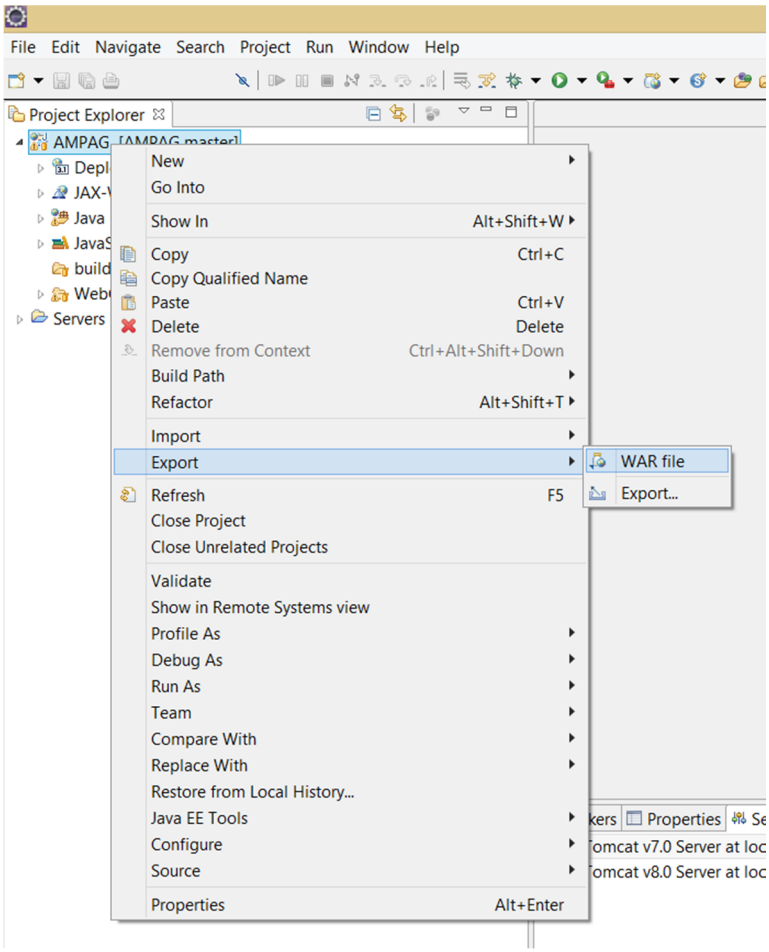

Ilustración 19 Eclipse - Exportar Fichero WAR 1

En la siguiente pantalla simplemente elegiremos en “Destination” la ruta donde queremos guardar el fichero WAR que nos creará Eclipse y pulsaremos el botón “Finish” dejando el resto de opciones a sus valores por defecto y ya tendremos en el destino elegido el fichero creado.

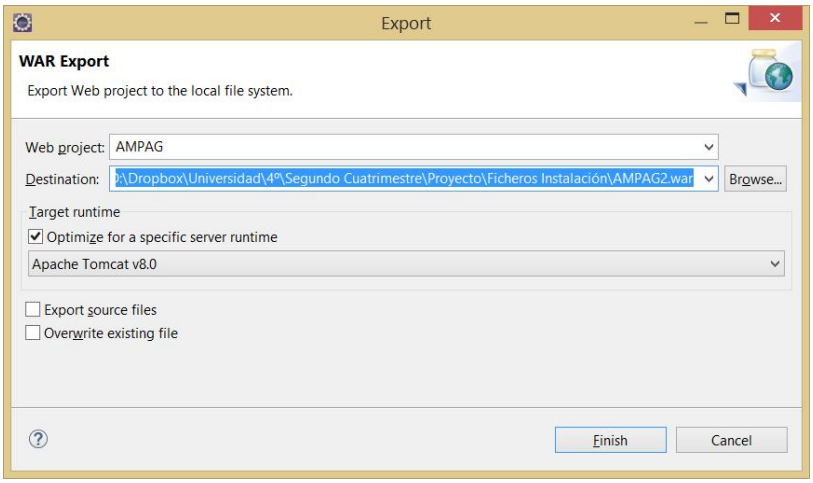

Ilustración 20 Eclipse - Exportar Fichero WAR 2

4.5 Estructura de Ficheros del Proyecto

Este apartado explica cómo se ha estructurado el proyecto en cuanto a todo el entramado de ficheros que contiene.

Existe una primera división entre los recursos JAVA (Java Resources) y el contenido web (WebContent) por tanto, cuando queramos mirar código correspondiente a las capas de modelo o controlador acudiremos a las carpetas dentro de los recursos Java y cuando deseemos ver parte de la capa vista iremos a las carpetas dentro del contenido web.

Ilustración 21 Estructura Proyecto - Código Capas

4.5.1 Recursos Java

Dentro de los recursos Java se encuentran todas las clases relacionadas con las capas de modelo y controlador de la aplicación. Para una mayor comprensión y organización se han dividido estas clases a su vez en distintos paquetes. De esta manera resulta muy sencillo para cualquier desarrollador coger el proyecto y entender como está organizado de un vistazo rápido. A continuación explicamos cada uno de los paquetes creados.

- **Beans**: Paquete con todas las clases que representan las tablas del modelo de datos

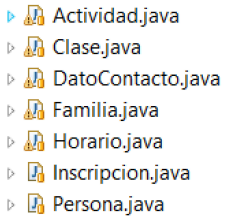

Ilustración 22 Estructura Proyecto - Paquete Beans

- **DAO**: Paquete con las clases que representan los DAOs para cada una de las tablas de la base de datos.

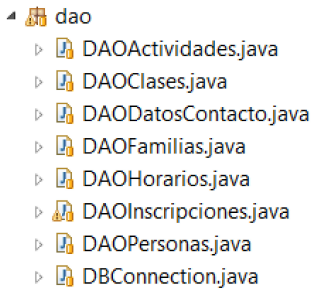

Ilustración 23 Estructura Proyecto - Paquete DAO

- **Exception**: Paquete que contiene clases para manejar excepciones.
- **Filter**: Paquete con la clase que realiza el filtro de datos de sesión en las peticiones HTTP. Aquí se añadirán más filtros si fuera necesario.
- **Reports**: Contiene las clases que generan los listados de la aplicación que se hacen con Dynamic Reports.

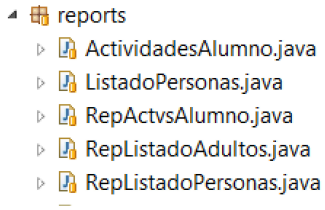

Ilustración 24 Estructura Proyecto - Paquete Reports

- **Servlets**: Este paquete contiene todas las clases de Servlet que se han creado. Se ha subdividido en cuatro paquetes más.
    - **Alta**: Servlets que realizan operaciones de inserción en la base de datos.
    - **Delete**: Servlets que realizan operaciones de eliminación en la base de datos.
    - **Edit**: Servlets que realizan operaciones de edición (operaciones UPDATE) en la base de datos.
    - **Get**: Servlets que rescatan datos de la base de datos (operaciones SELECT).

  Luego además hay otros servlets no incluidos en ninguno de esos paquetes que realizan funciones muy específicas.

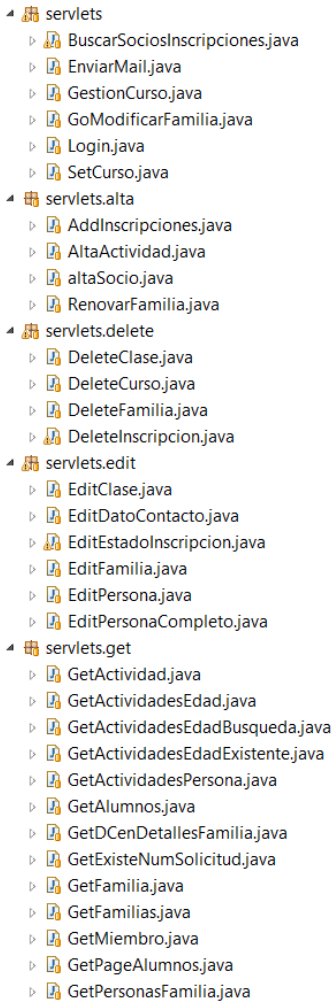

Ilustración 25 Estructura Proyecto - Paquetes Servlets

- **Utils**: Contiene clases para realizar funciones muy específicas como por ejemplo el envío de emails.

4.5.2 Contenido Web

Dentro de la carpeta WebContent se encuentran diseminados todos los ficheros tanto de páginas JSP como ficheros JavaScript y de hojas de estilo (CSS) que forman finalmente toda la estructura de la capa vista de la aplicación. Para una mayor comprensión y organización se han dividido estas clases a su vez en distintas carpetas en función su formato. De esta manera nos encontramos con las siguientes carpetas.

- **CSS**: Contiene todas las hojas de estilo así como las imágenes que usa la aplicación. Aquí se encuentran los estilos de bootstrap, de los controles personalizados, de jqGrid, jQuery, etc.

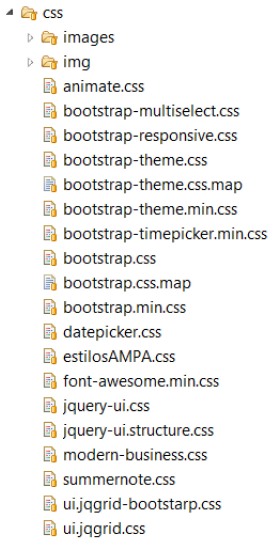

Ilustración 26 Estructura Proyecto - Carpeta CSS

- **Fonts**: Ficheros con todos los tipos de fuente de texto que usa la aplicación.

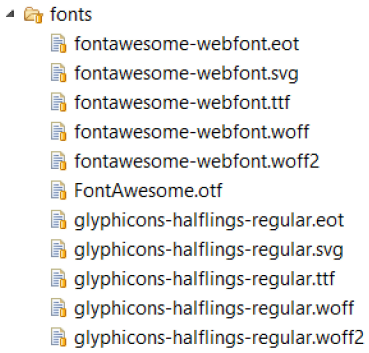

Ilustración 27 Estructura Proyecto - Carpeta Fonts

- **Js**: Esta carpeta incluye todos los ficheros de tipo JavaScript que se utilizan en la aplicación, desde los requeridos por Bootstrap, jQuery y jqGrid hasta los creados específicamente para este proyecto.

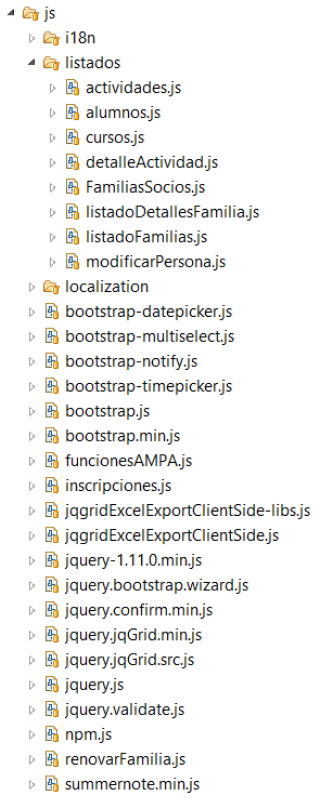

Ilustración 28 Estructura Proyecto - Carpeta JS

- **Pages**: Incluye todas las páginas JSP diseñadas durante el desarrollo de la aplicación.

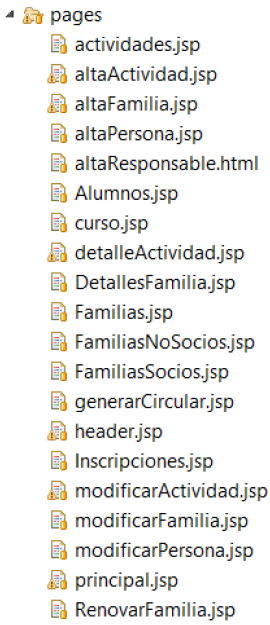

Ilustración 29 Estructura Proyecto - Carpeta Pages

- **Plugins**: Ficheros JavaScript que funcionan a modo de Plugins de jqGrid y Bootstrap.

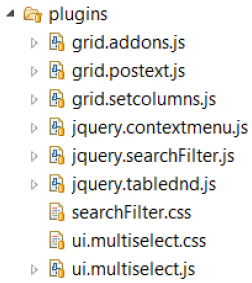

Ilustración 30 Estructura Proyecto - Carpeta Plugins

- **WEB-INF**: Directorio que contiene todos los recursos relacionados con la aplicación web que no se han colocado en el directorio raíz y que no deben servirse al cliente. Esto es importante, ya que este directorio no forma parte del documento público, por lo que ninguno de los ficheros que contenga va a poder ser enviado directamente a través del servidor web. En este directorio se coloca el archivo web.xml, donde se establece la configuración de la aplicación web o las librerías usadas por la aplicación.

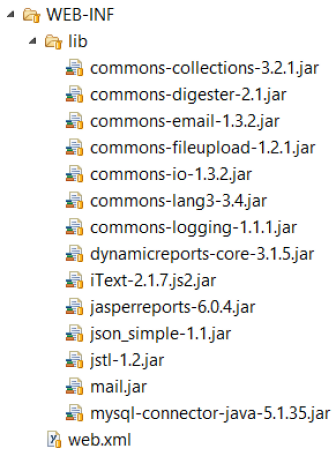

Ilustración 31 Estructura Proyecto - Carpeta WEB-INF

Capítulo 5
5 Manual de instalación

Este apartado muestra los pasos a realizar para tener la aplicación desplegada y funcionando en un ordenador.

Para la instalación se utilizarán los ficheros proporcionados en el CD incluido junto la documentación de este proyecto. En concreto hay que realizar la instalación de cuatro componentes software, el JDK de JAVA, Apache Tomcat 7, MySQL 5 y desplegar el fichero WAR dentro de Apache Tomcat.

Todo el entorno se explica para un ordenador con sistema operativo Windows debido a su presencia mayoritaria en la mayoría de los hogares pero se recuerda que el proyecto es multiplataforma y podría realizarse la instalación de los mismos componentes en otros sistemas operativos. Además en el CD de instalación se incluyen las versiones de 32 bits y de 64 bits para aquellos programas que cuentan con distintos ficheros de instalación.

5.1 Instalación del JDK 8 de JAVA

Para realizar esta instalación se utilizará el fichero “jdk-8u45-windows-x64” para Windows de 64 bits o bien el fichero “jdk-8u45-windows-i586” para las versiones de 32 bits.

La instalación es bastante sencilla y es seguir todos los pasos del asistente eligiendo las opciones de configuración por defecto. Una vez instalado, se pasa a verificar su correcta configuración. Para ello habrá que comprobar que la variable de entorno JAVA_HOME se ha creado correctamente en las variables de entorno del sistema. Para ello hay que acceder a las variables de entorno que se encuentran ubicadas en el Panel de Control en la opción “Sistema”, “Configuración Avanzada del Sistema” y finalmente en el pop up resultante pulsamos el botón “Variables de Entorno”.

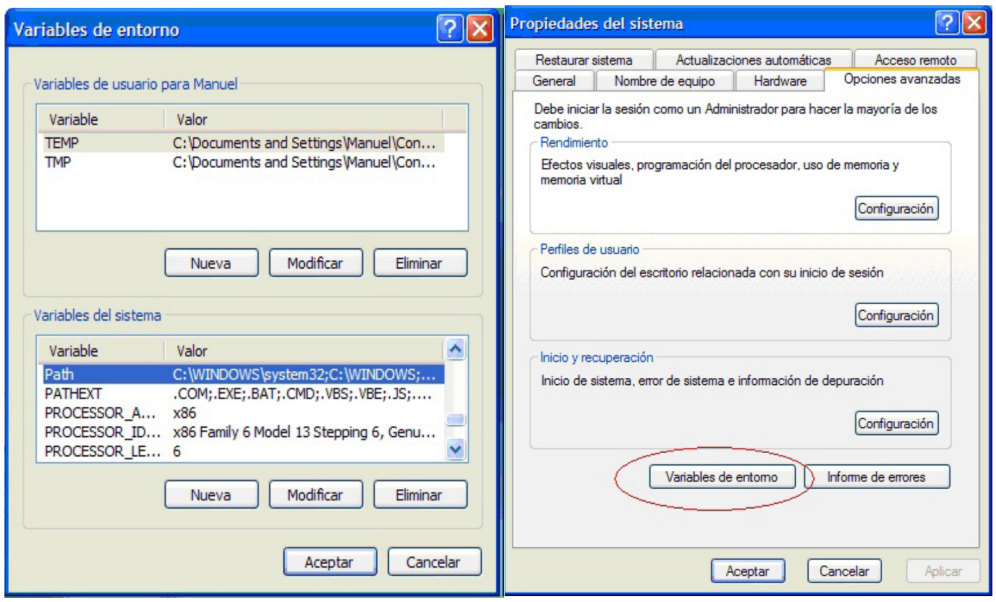

Ilustración 32 Comprobación Variables de Entorno

5.2 Instalación de Apache Tomcat 8

Se utilizará el fichero “apache-tomcat-8.0.23” siendo este válido tanto para versiones 32 bits como para 64 bits.

En la ventana de Bienvenida se pulsa Next para empezar con la instalación:

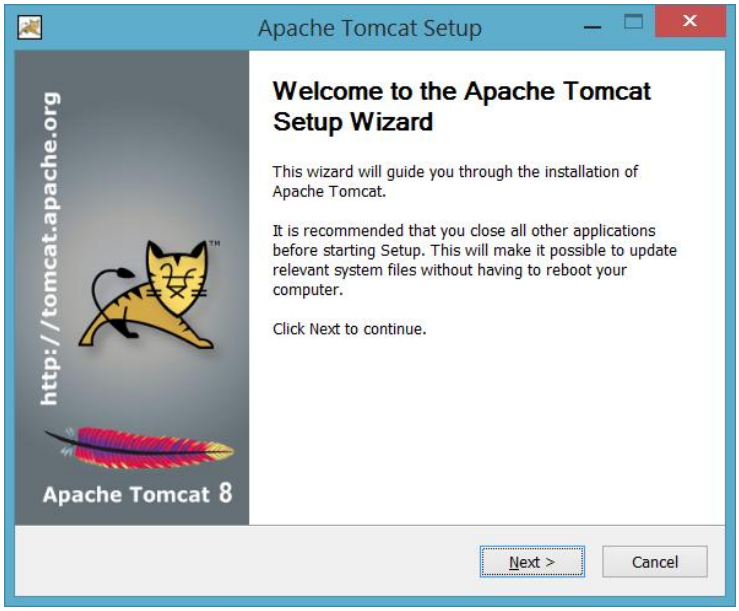

Ilustración 33 Instalación Apache Tomcat - Pantalla Bienvenida

En la siguiente ventana se muestra el acuerdo de licencia, el cual se debe aceptar para seguir con la instalación, para ello se pulsa el botón I Agree:

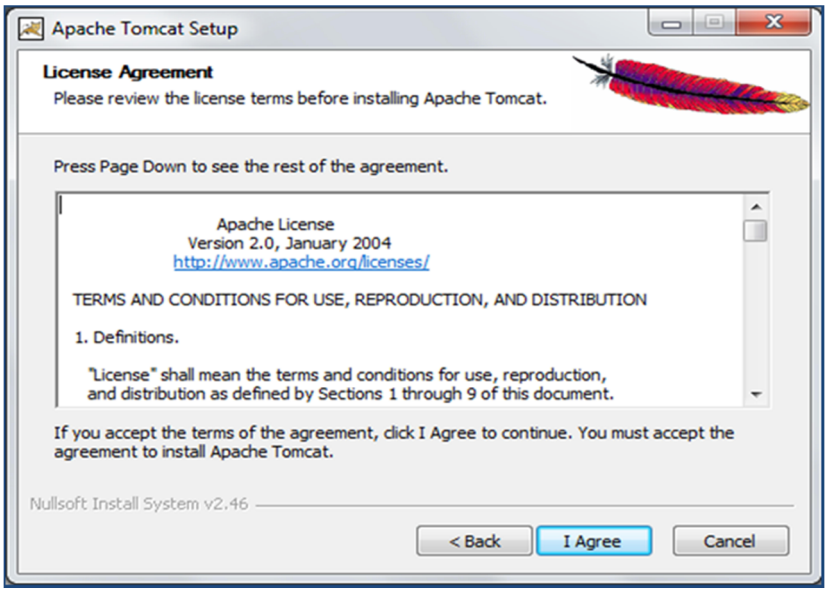

Ilustración 34 Instalación Apache Tomcat - Pantalla Licencia Usuario

En el siguiente paso se procede a seleccionar el tipo de instalación que se quiere hacer, para esta práctica se selecciona Normal del desplegable, que es la opción por defecto; los componentes a instalar dependerán del tipo de instalación seleccionada, pulsamos Next para continuar:

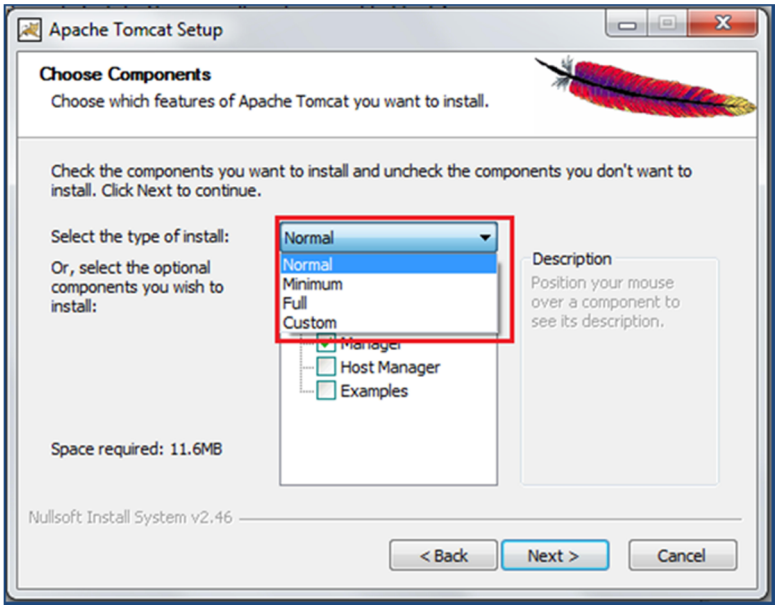

Ilustración 35 Instalación Apache Tomcat - Selección tipo de instalación

En la siguiente ventana se pueden modificar distintas opciones de configuración, se dejan los puertos que vienen por defecto, se asigna un nombre de usuario (admin) y una contraseña (admin) y pulsamos Next:

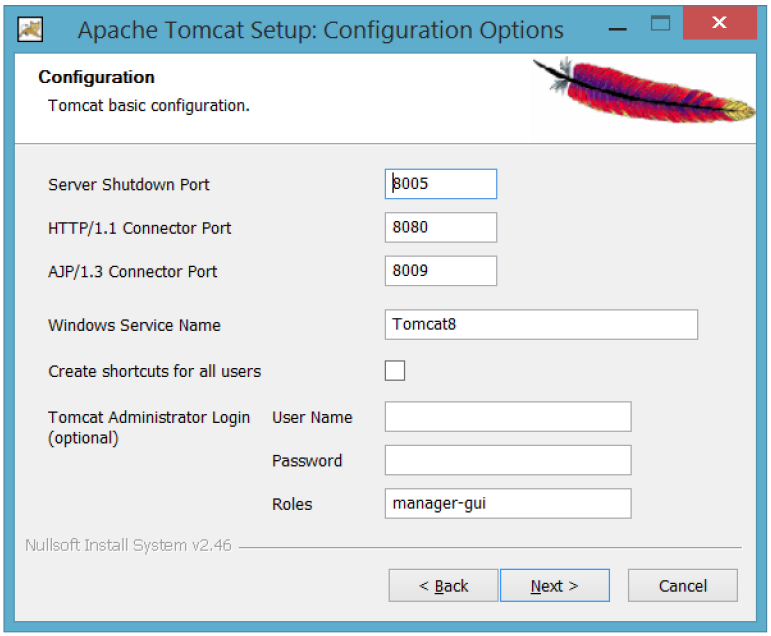

Ilustración 36 Instalación Apache Tomcat – Configuración de instalación

Apache Tomcat necesita para poder ejecutarse correctamente la máquina virtual de Java, en la siguiente ventana se le puede indicar la ruta donde está instalada; en este caso, si la instalación de Java fue realizada con la configuración por defecto como se ha indicado, Apache encuentra la ruta donde está instalada en la ruta por defecto. Se pulsa Next para continuar:

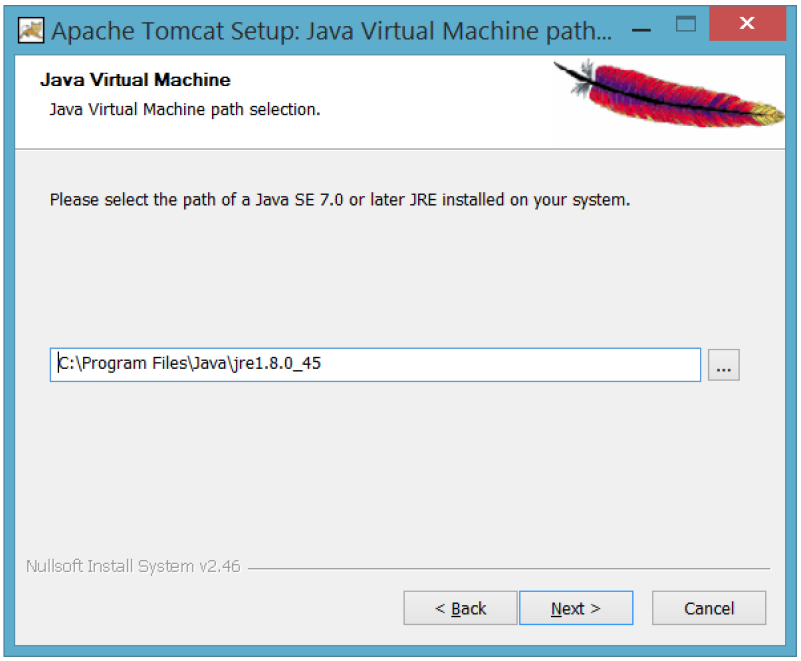

Ilustración 37 Instalación Apache Tomcat – Ruta de instalación de JAVA

La última opción antes de comenzar la instalación de Apache Tomcat es indicar la ruta donde se quiere instalar, se recomienda la ruta C:\Apache.

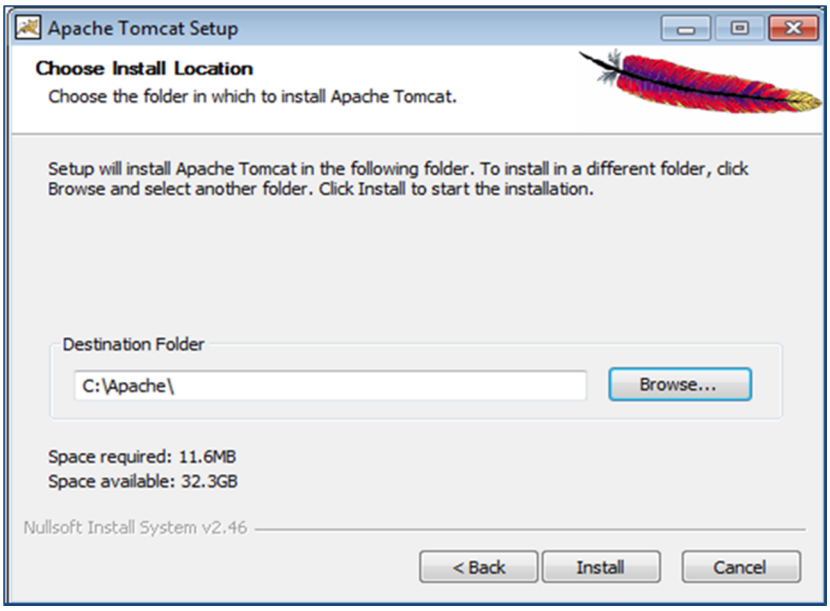

Ilustración 38 Instalación Apache Tomcat – Ruta de instalación

Una vez terminada la instalación, se pulsa Finish para arrancar Apache Tomcat:

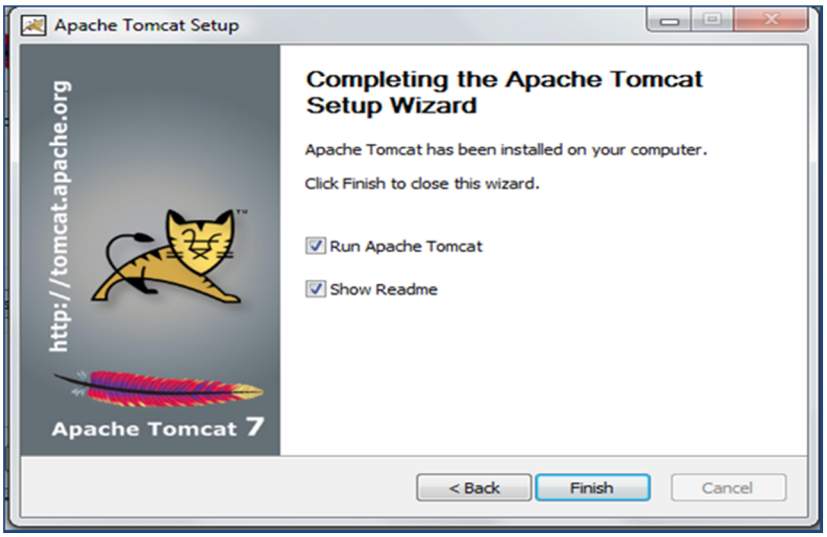

Ilustración 39 Instalación Apache Tomcat – Instalación Finalizada

Aparece una ventana que muestra el progreso de arranque del servicio:

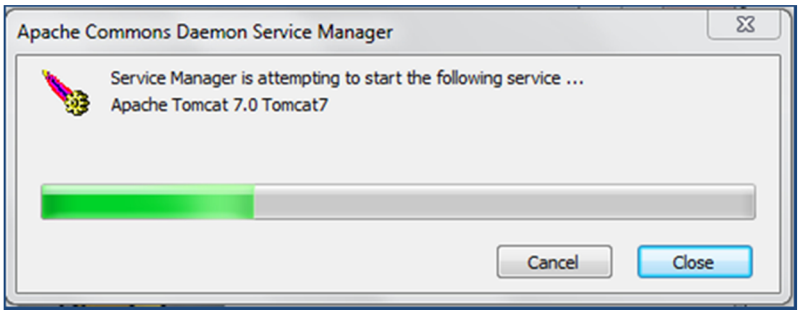

Ilustración 40 - Instalación Apache Tomcat - Inicialización del servicio

Una vez el sistema se haya iniciado se puede observar como en la barra de herramientas en segundo plano hay un nuevo icono correspondiente a la herramienta “Monitor Apache” que indica con una flecha verde que el Apache Tomcat está iniciado.

Ilustración 41 Instalación Apache Tomcat - Icono Segundo Plano

Para evitar que durante la ejecución dela práctica surjan problemas debido a que los puertos que utiliza Apache estén ya siendo utilizados, se deberá parar el servidor ya que será Eclipse el que lo inicie automáticamente.

5.3 Instalación de MySQL

Se utilizará el fichero “mysql-installer-community-5.5.44.0” siendo este válido tanto para versiones 32 bits como para 64 bits.

La primera pantalla será el acuerdo de licencia el cual deberemos de aceptar para poder pulsar el botón “Next”.

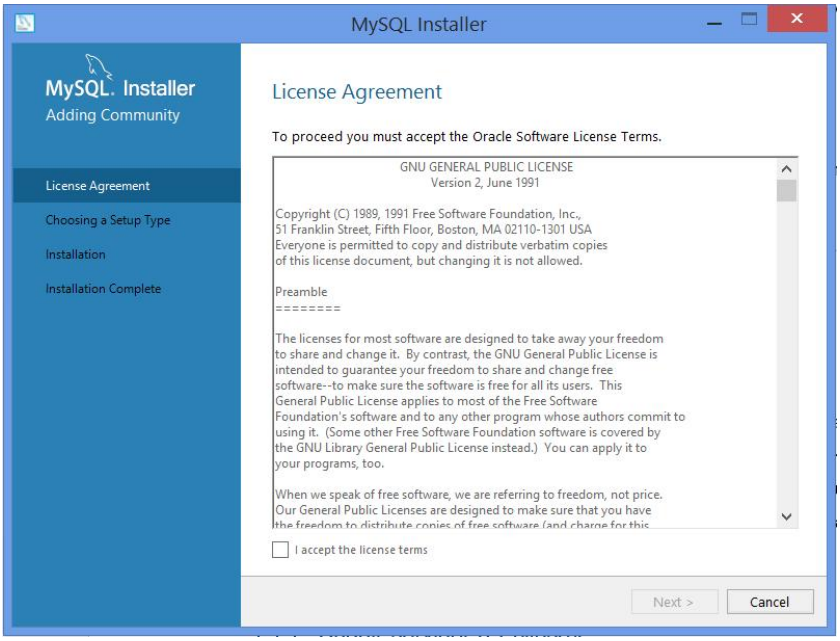

Ilustración 42 Instalación MySQL - Acuerdo de Licencia

Continuaremos seleccionando el tipo de instalación que queremos llevar a cabo. Elegiremos la opción “Custom” que es personalizada y pulsaremos el botón “Next”.

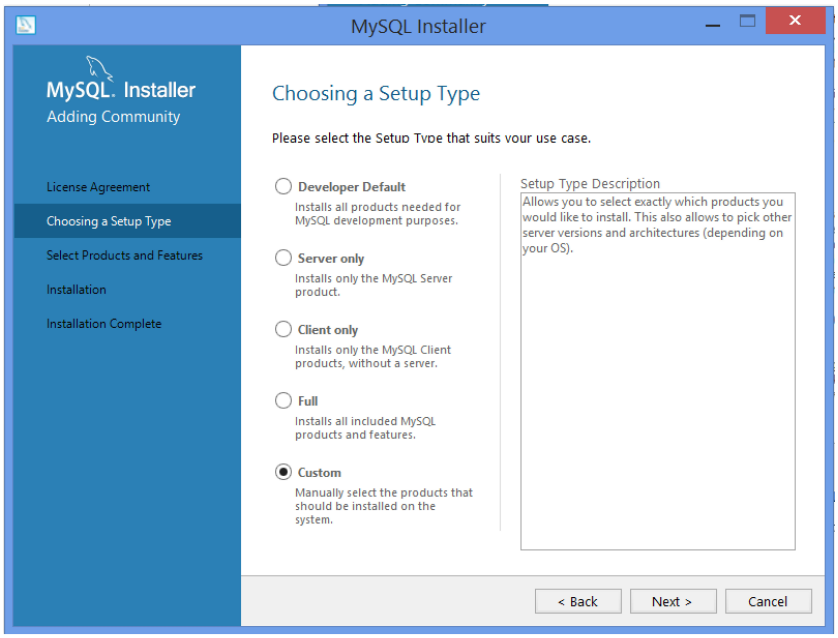

Ilustración 43 Instalación MySQL - Tipo de Instalación

A continuación deberemos elegir los componentes que queremos instalar. Se seleccionará la versión de MySQL Server x86 en sistemas de 32bit y la de x64 en los de 64 bits. Además incluiremos el conector JDBC. Continuaremos la instalación pulsando el botón “Next”.

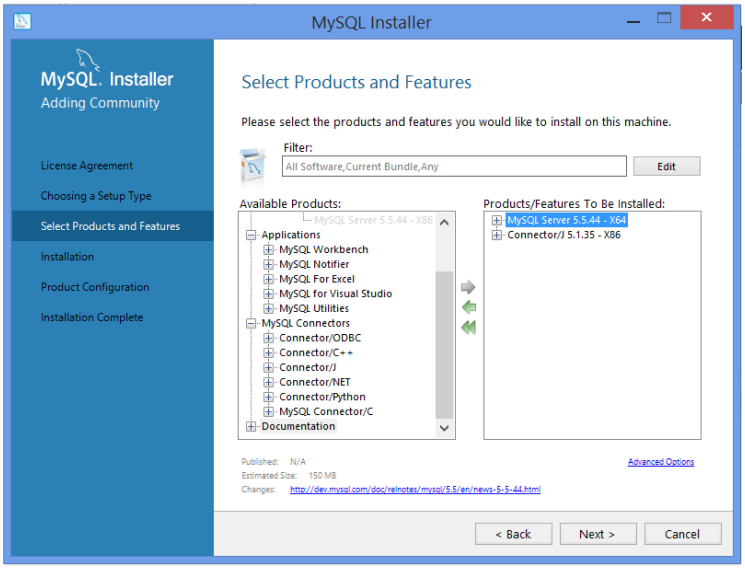

Ilustración 44 Instalación MySQL - Selección de componentes

El asistente de instalación mostrará un resumen de lo que se va a instalar, se continuará la instalación pulsando el botón “Execute”.

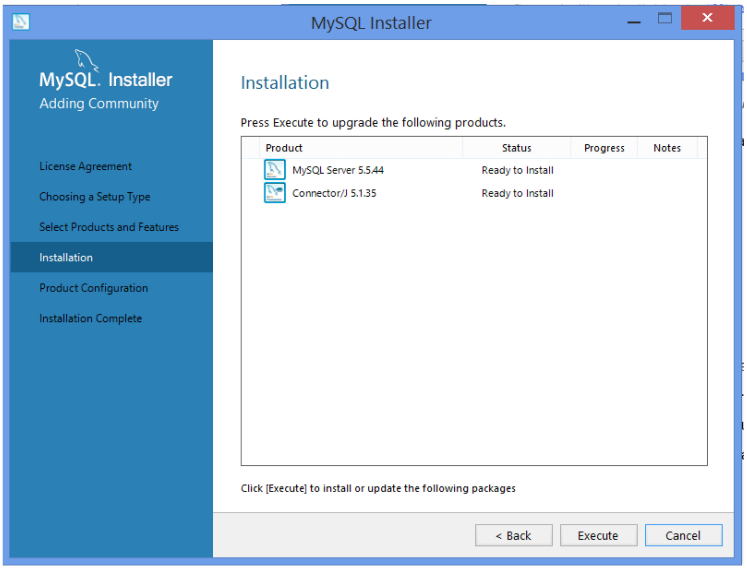

Ilustración 45 Instalación MySQL - Resumen Instalación seleccionada

La instalación habrá finalizado una vez todos los componentes muestren su status como "Complete". Se continuará la instalación pulsando el botón "Next".

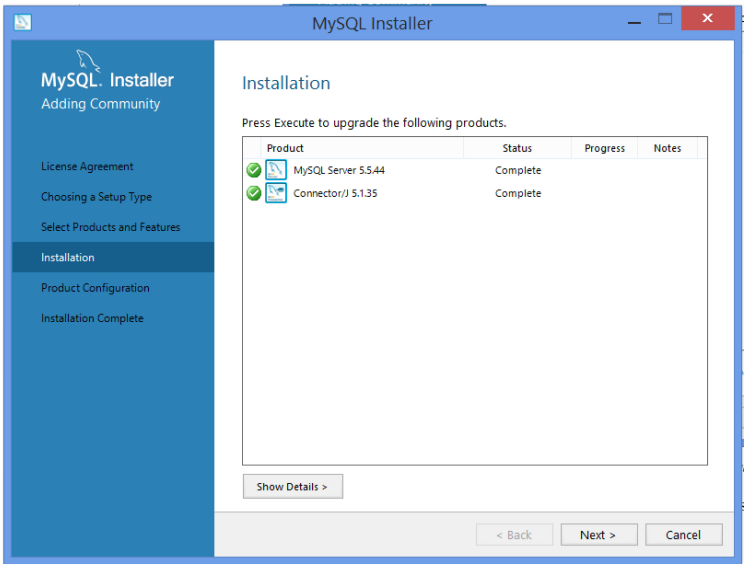

Ilustración 46 Instalación MySQL - Instalación de componentes completada

El asistente procederá ahora a indicar que MySQL está preparado para configurarse y que tras pulsar el botón "Next" va a guiar al usuario a configurarlo.

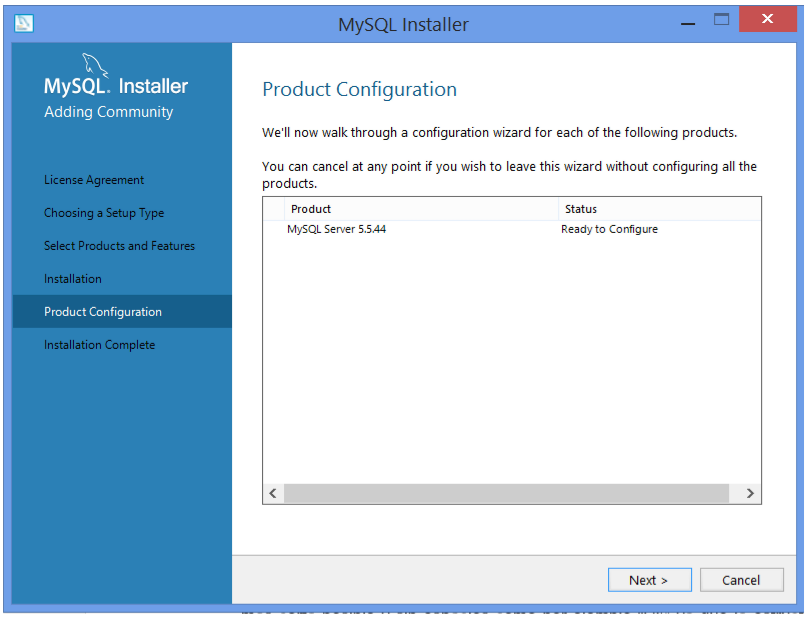

Ilustración 47 Instalación MySQL - Comenzando la configuración

La primera pantalla de configuración será la selección del tipo de servidor MySQL que se está configurando. Se selecciona la opción “Server Machine”, se deja el resto de opciones a sus valores por defecto y se pulsa el botón “Next”.

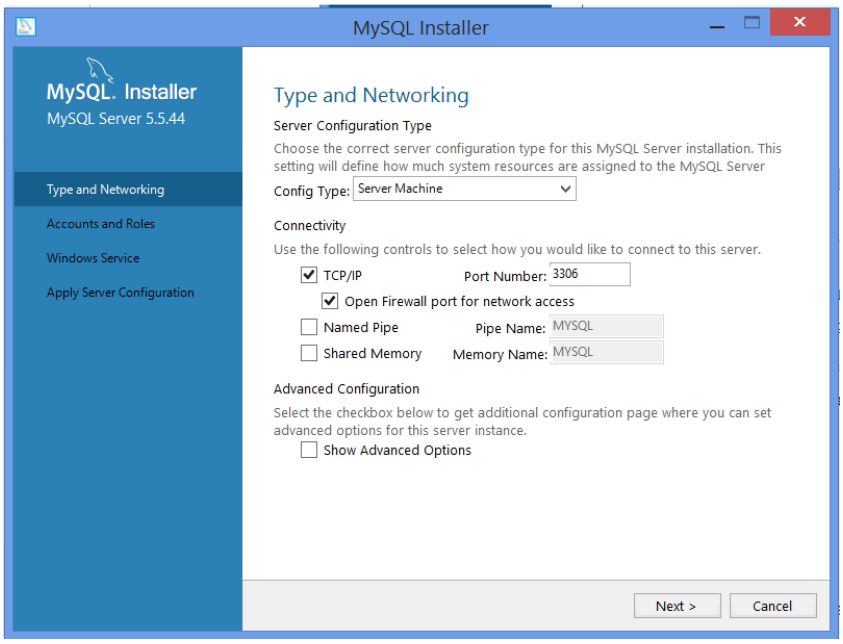

Ilustración 48 Instalación MySQL - Configuración del MySQL 1

El siguiente paso es configurar la contraseña del usuario “root” de MySQL. Se establece una contraseña y se deja el resto sin modificar y se continúa pulsando el botón “Next”.

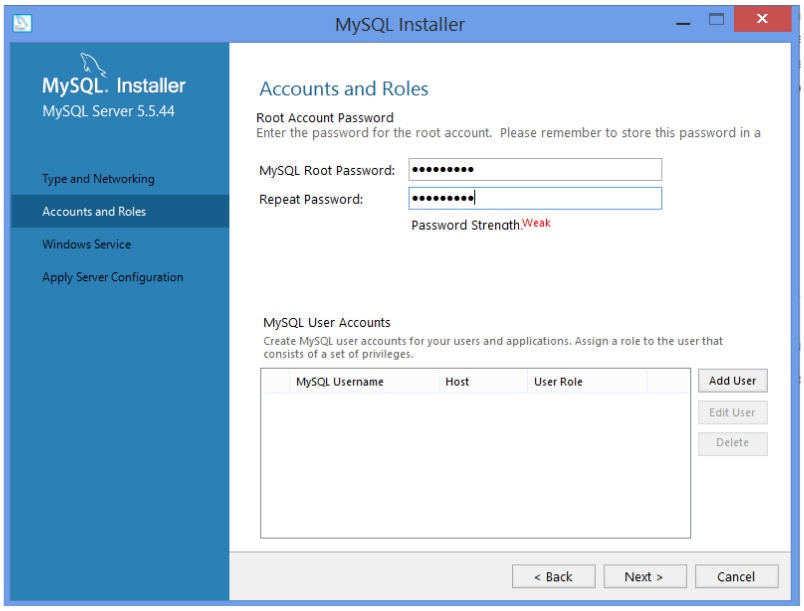

Ilustración 49 Instalación MySQL - Configuración del MySQL 2

La siguiente pantalla configura MySQL para que funcione como un servicio de Windows y que se ejecute automáticamente al iniciar Windows. No se modificar nada en esta pantalla y simplemente se continúa pulsando el botón “Next”.

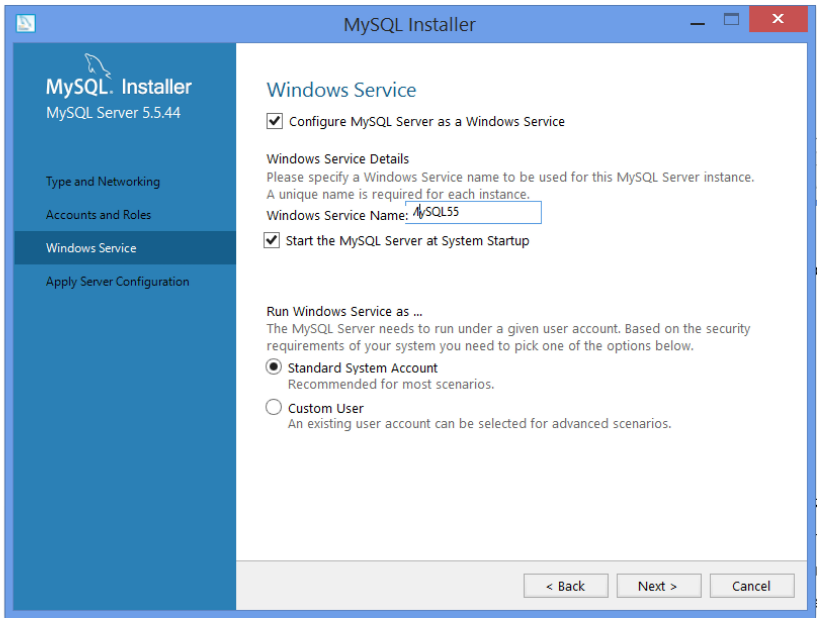

Ilustración 50 Instalación MySQL - Configuración del MySQL 3

La siguiente pantalla pide la confirmación del usuario para comenzar a aplicar los cambios de configuración seleccionados. Se pulsa el botón “Execute” y se espera a que finalice.

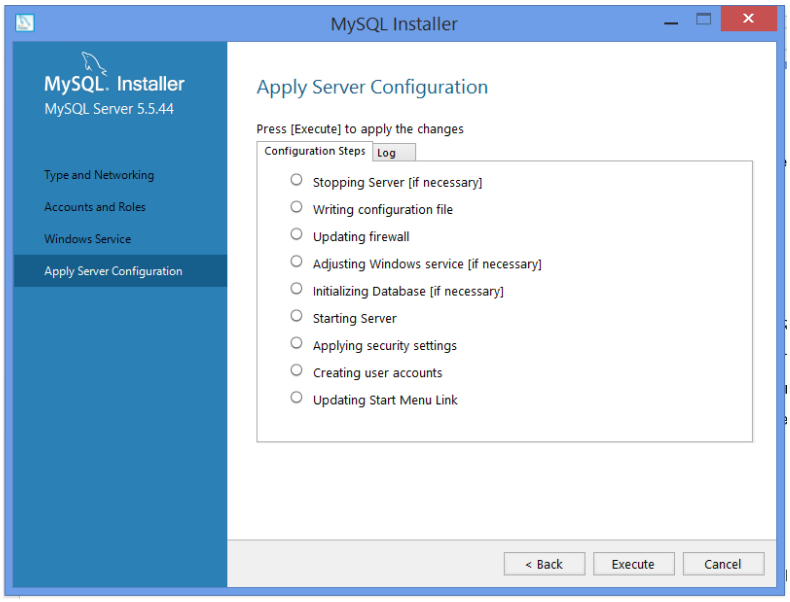

Ilustración 51 Instalación MySQL - Configuración del MySQL 4

Una vez todos los pasos se hayan completado el asistente nos mostrará habilitado el botón de "Finish" para que podamos finalizar el asistente de instalación. La instalación de MySQL se habrá completado exitosamente.

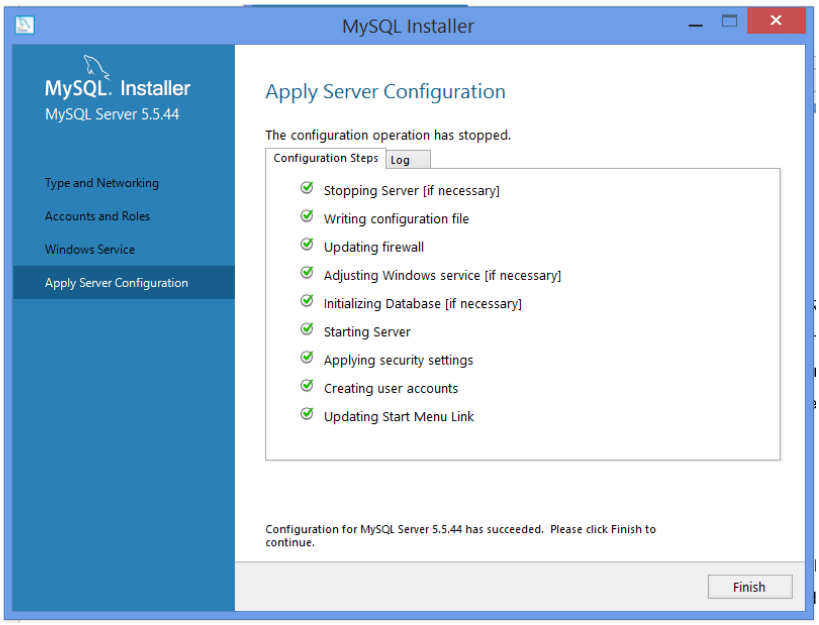

Ilustración 52 Instalación MySQL - Instalación finalizada

5.3.1 Configuración de la Base de Datos

Tras la instalación de MySQL hay que proceder a ejecutar el script de configuración de la base de datos que procederá a crear dicha base de datos con todas sus tablas y procedimientos y también un usuario con acceso a ella. Los datos de la base de datos y del usuario son:

- **Nombre de la base de datos**: ampa_garena
- **Nombre de usuario**: ampa
- **Contraseña**: ampa.2K15

Para la creación de la base de datos y del usuario habrá que lanzar el script "scriptConfiguracionBD.sql" incluido en el CD de instalación dentro de la carpeta MySQL. Para lanzarlo bastará con abrir el "MySQL 5.5 Command Line Client" que se encuentra ubicado en el menú de inicio de Windows (si no estuviera habría que lanzar el comando "C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe" "--defaults-file=C:\ProgramData\MySQL\MySQL Server 5.5\my.ini" "-uroot" "-p") introducir el password de root que hemos configurado durante la instalación de MySQL y ejecutar el siguiente comando:

source rutaScriptConfiguración

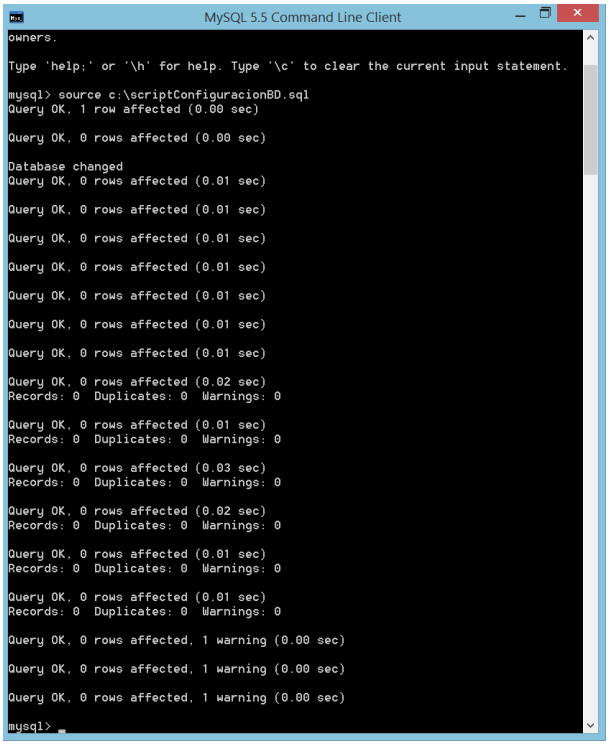

Ilustración 53 Configuración MySQL - Creación de la base de datos y el usuario

Una vez terminada la ejecución de la instrucción, MySQL estará correctamente configurado para funcionar con la aplicación.

5.4 Despliegue de la Aplicación

Una vez instalados todos los componentes necesarios para ejecutar la aplicación hay que proceder a realizar su instalación por medio del despliegue del fichero WAR en Apache Tomcat.

Se utilizará el fichero WAR "ActividadesExtraescolaresAMPAG.war" incluido en el CD de instalación anexo a este proyecto.

El primer paso será copiar el fichero WAR en la carpeta "webapps" del directorio donde se haya instalado Apache Tomcat. Por defecto la ruta sería "C:\Program Files\Apache Software Foundation\Tomcat 8.0\webapps".

El siguiente paso será detener y reanudar el servicio de Apache Tomcat para ello se accede mediante clic derecho al icono de Apache Tomcat de la barra de tareas en segundo plano y se hace clic en "Stop Service". Acto seguido repetimos la misma acción pero ahora haremos clic en "Start Service".

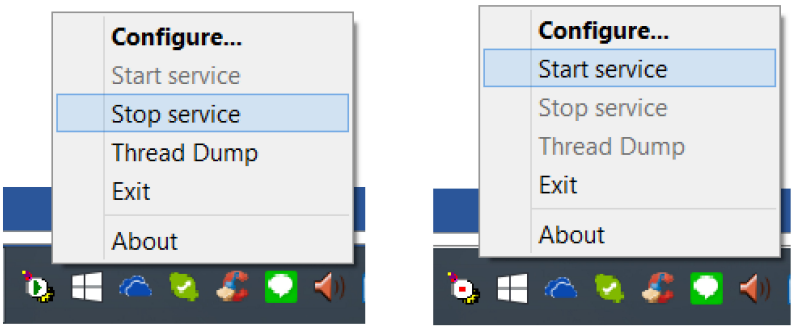

Ilustración 54 Reiniciar Apache Tomcat

Si todo ha ido correctamente se puede ver como en la carpeta "webapps" de Apache Tomcat ahora ha aparecido un nuevo directorio llamado "AMPAG" que contiene todo el árbol de directorios de la aplicación. Por tanto ahora si se accede a la dirección http://127.0.0.1:8080/AMPAG/ aparecerá la pantalla de acceso de la aplicación.

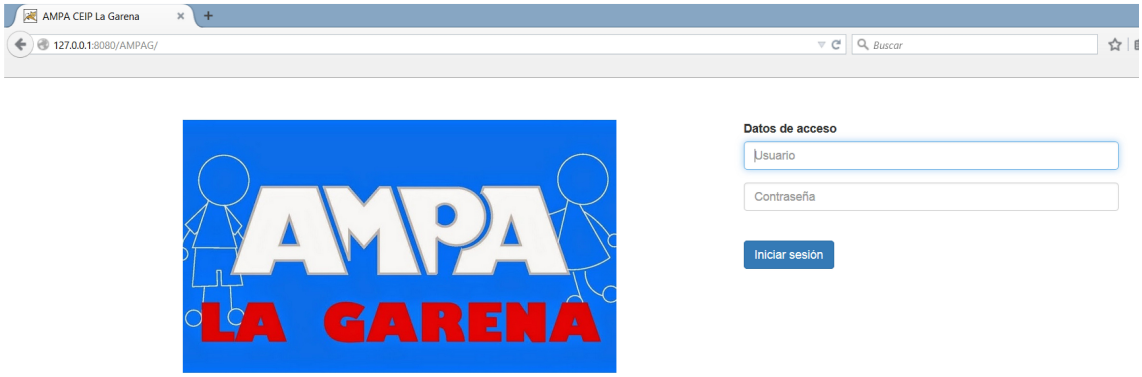

Ilustración 55 Pantalla Bienvenida de la aplicación

Capítulo 6
6 Manual de Usuario

En este apartado se detalla el funcionamiento de la aplicación desde el punto de vista del usuario y como realizar todas las funciones disponibles.

6.1 Consideraciones Previas

Este manual de uso está estructurado por secciones entre las distintas pantallas disponibles en la aplicación. Para cada sección se presentará una imagen con globos numerados y a continuación cada uno de esos globos explicando la función o detalle que señalan.

6.2 PANTALLA PRINCIPAL

Ilustración 62 Manual de Usuario – Pantalla Principal

1. Desde este apartado el usuario puede generar todos los posibles listados PDF que se permiten en la aplicación simplemente haciendo clic en el listado que desee crear.
2. Para cambiar el curso con el que el usuario quiere trabajar.
3. Sección de accesos directos a páginas de la aplicación.

Ilustración 56 Manual de Usuario – Uso del manual

Además previamente se explican apartados generales que se repiten a lo largo de las secciones para evitar extender en exceso el manual. Si en algún caso las secciones generales contase con algún apartado o detalle extra en alguna sección, este detalle se explicará en la sección correspondiente.

6.1.1 Acceso a la Aplicación

Para acceder a la aplicación se debe escribir en la barra de direcciones del navegador Web la URL http://localhost:8080/AMPAG/.

En la pantalla que se muestra al usuario, éste debe estar registrado previamente, e introducir su usuario y contraseña de acceso en el formulario.

Ilustración 57 Manual de Usuario - Pantalla de Login

6.1.2 Menú

A lo largo de todas las páginas acompaña un menú de cabecera que permite acceder a todas las páginas de la aplicación.

Ilustración 58 Manual de Usuario – Menú

1. Enlace a la página principal. Además indica el curso actual con el que se está trabajando.
2. Sección Familias. Al hacer clic se despliega el submenú con las acciones relativas a las familias.
3. Ejemplo de submenú. Al hacer clic en alguna de sus opciones la aplicación cargará la página correspondiente a la acción señalada por el submenú.
4. Sección Actividades. Al hacer clic se despliega el submenú con las acciones relativas a las actividades.
5. Sección Alumnos. Al hacer clic se despliega el submenú con las acciones relativas a los alumnos.
6. Sección Otros. Al hacer clic se despliega el submenú con las acciones para generar nuevos cursos o una circular de correo
7. Botón Salir. Cierra la sesión del usuario, limpia variables y carga de nuevo la página de login.

6.1.3 Notificaciones Emergentes

Cuando en la aplicación el usuario realiza una acción que necesita una confirmación, por ejemplo dar de alta una familia, para saber si se ha realizado correctamente, la aplicación muestra unas notificaciones emergentes con el resultado obtenido por la aplicación. Estas notificaciones pueden indicar que la acción se realizó correctamente y mostrarán una notificación verde o bien la notificación puede señalar un error al realizar la acción y mostrar una notificación roja.

Ilustración 59 Manual de Usuario – Notificaciones Emergentes

6.1.4 Tablas

En la mayoría de las pantallas de la aplicación se utilizan tablas, ya sea para mostrar una lista de participantes en una actividad, para mostrar todas las familias registradas o cualquier otro tipo de situación. A continuación se explican todas las posibilidades que permiten las tablas para que el usuario sepa utilizarlas cuando se encuentre con ellas ya que tienen la capacidad de permitir muchas acciones.

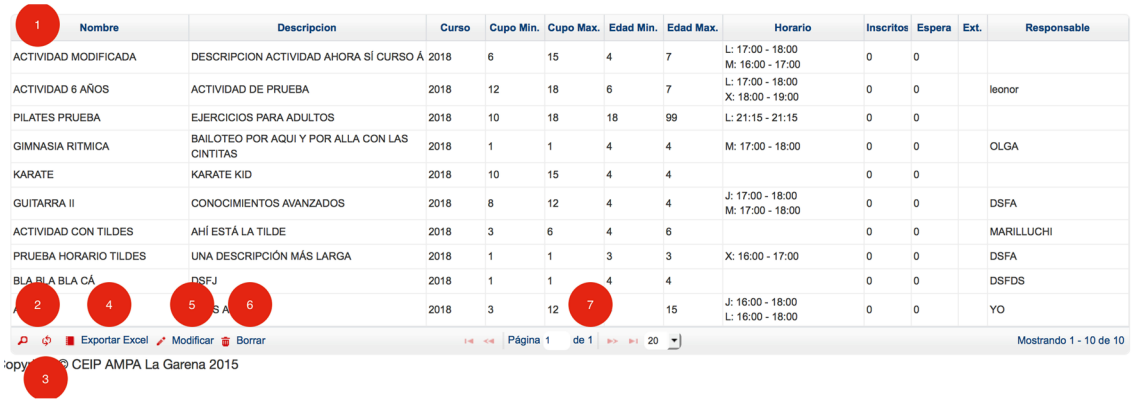

Ilustración 60 Manual de Usuario – Tablas

1. La primera fila de cada tabla representa los nombres del tipo de dato que representa cada columna. Haciendo un clic sobre cualquier columna ordena la tabla en función de esa columna con un orden ascendente y con un segundo clic la ordenación pasará a ser descendente.
2. El icono de la lupa permite realizar una búsqueda sobre la tabla. Al hacer clic sobre él aparece un formulario en forma de pop up donde nos solicita en base a que campos

queremos buscar y que valores. Tras establecer los criterios se pulsa el botón buscar y la tabla realizará la búsqueda devolviendo o no valores.

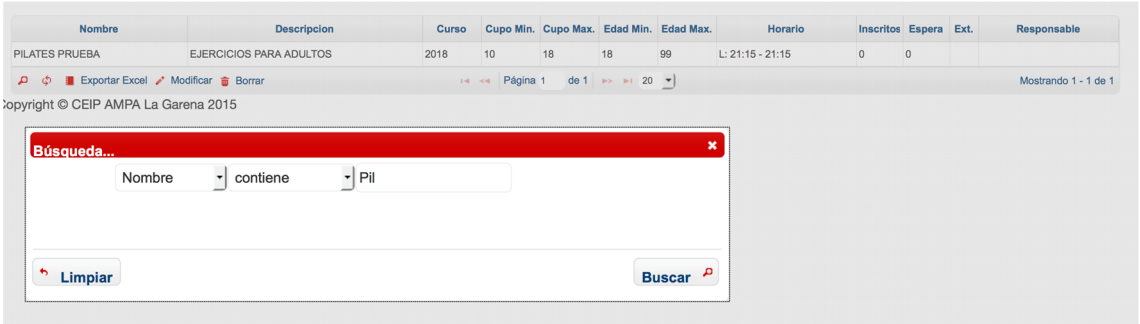

Ilustración 61 Manual de Usuario – Búsqueda en tablas

3. Este icono permite reiniciar la tabla a su estado inicial en caso de que hayamos realizado una búsqueda.
4. Este botón permite realizar un fichero Excel que se descargará automáticamente en el ordenador con el contenido que este mostrando la tabla en ese instante.
5. El botón “Modificar” permite editar la información de la fila que esta seleccionada en la tabla y por consiguiente, en la base de datos de la aplicación. Al hacer clic se mostrará un pop-up con un formulario para introducir los nuevos datos. Esta opción puede no estar incluida en todas las tablas.

Ilustración 62 Manual de Usuario – Modficación en tablas

6. El botón “Borrar” elimina la fila seleccionada de la tabla y de la base de datos de la aplicación. Esta opción puede no estar incluida en todas las tablas.
7. Controles de paginación. Permiten desplazarse por las distintas páginas que contenga la tabla. Es posible seleccionar que una tabla muestre más o menos filas por página por medio del control situado más a la derecha.

6.2 Pantalla Principal

Ilustración 63 Manual de Usuario – Pantalla Principal

1. Desde este apartado el usuario puede generar todos los posibles listados PDF que se permiten en la aplicación simplemente haciendo clic en el listado que desee crear.
2. Para cambiar el curso con el que el usuario quiere trabajar.
3. Sección de accesos directos a páginas de la aplicación.

6.3 Menú Familias

La sección del menú de Familias tiene en su haber un total de 4 opciones de las cuales 3 son comunes, “Socios”, “No Socios” y “Ver Todos”; y la última opción permite añadir una nueva familia a la aplicación. A continuación se explican todas las pantallas de este menú y aquellas que se derivan de ir accediendo a través de las distintas opciones.

6.3.1 Socios, No Socios y Ver Todos

Estas tres opciones comunes y llevan a la misma página que es un listado de las familias que hay en la aplicación. Si se elige la opción “Socios” sólo se mostrará un listado con familias que son socias, si se elige la opción “No Socios” sólo se mostrará un listado con familias que no son socias y si por el contrario se elige la opción “Ver todos” se mostrará un listado con todas las familias que han sido registradas en la aplicación.

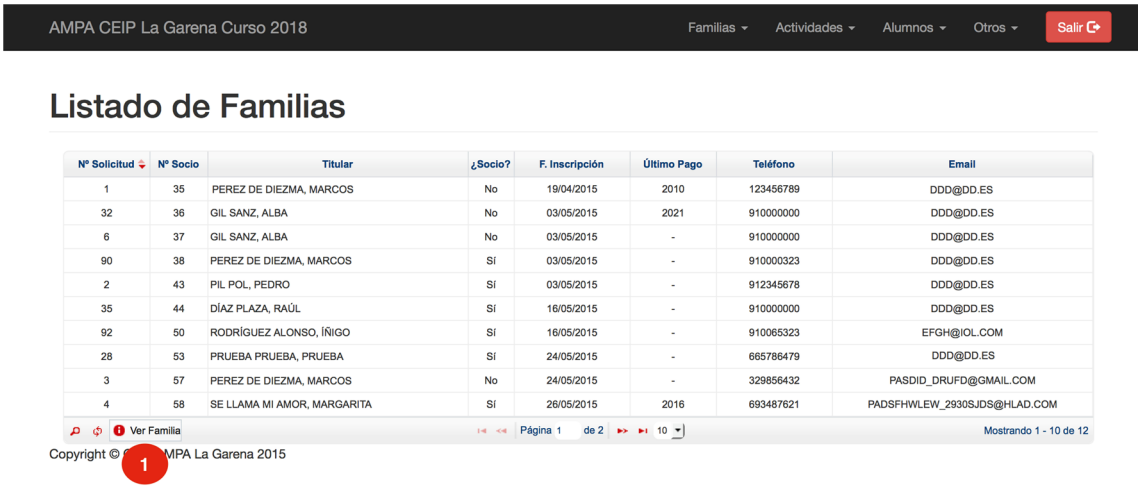

Ilustración 64 Manual de Usuario – Listado Familias

1. El botón “Ver Familia” permite ir a la página que muestra la información de la familia seleccionada.

6.3.2 Añadir Familia

Cuando el usuario acceda a la opción “Añadir Familia” se mostrará una página con un formulario solicitando la información básica necesaria para registrar a una familia en la aplicación y además se pide al usuario que introduzca a los miembros de la familia siendo necesario introducir al menos a un miembro. También se da la posibilidad de introducir los datos de contacto de la familia.

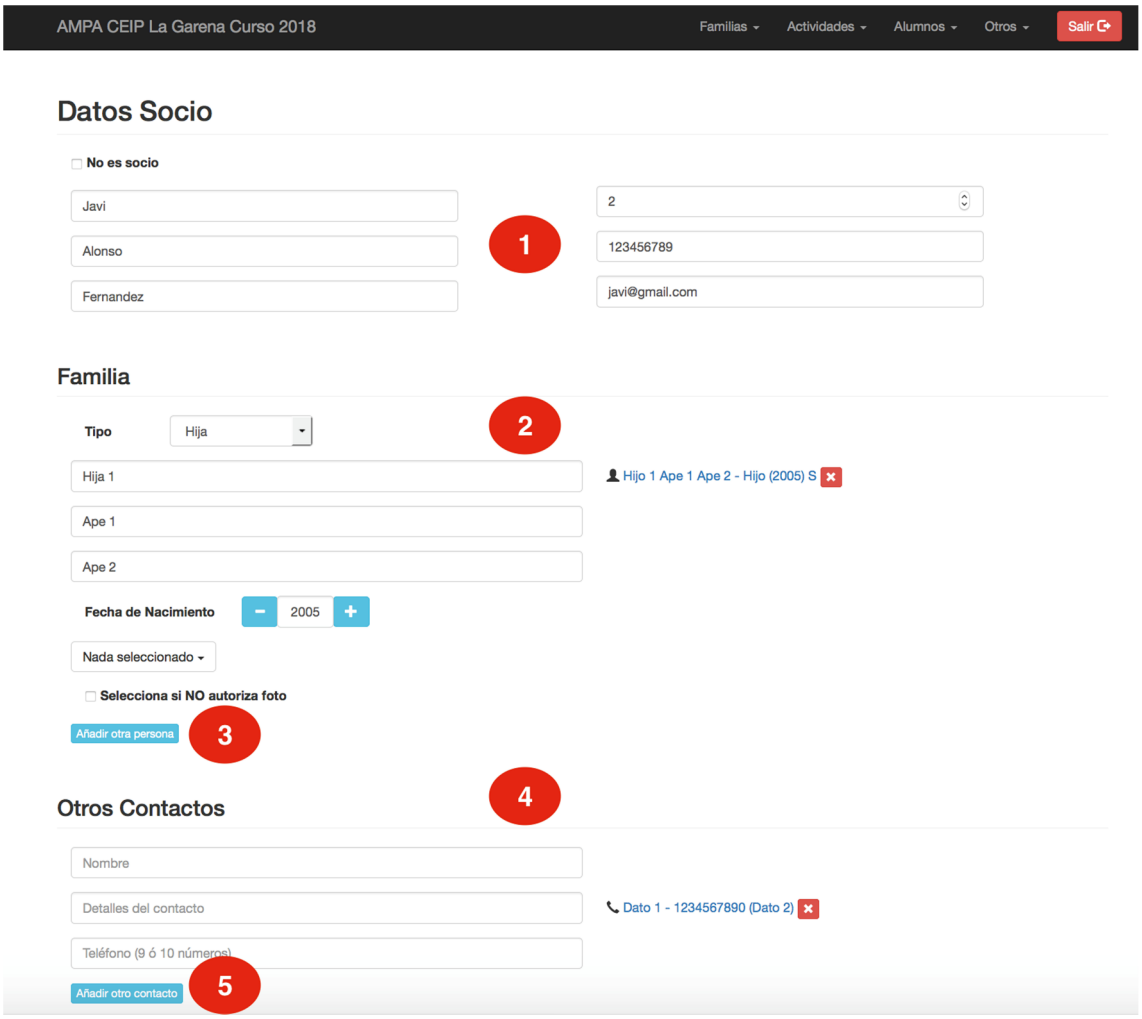

Ilustración 65 Manual de Usuario – Añadir Familia

1. Datos necesarios para registrar a una familia. La aplicación verifica que el usuario introduzca datos válidos para el teléfono (9 o 10 dígitos) y para el email (formato de email correcto) y además valida que el número de solicitud introducido no se encuentre ya registrado en la base de datos.
2. Datos necesarios para registrar a una persona. La aplicación verifica que el usuario introduzca datos válidos para el teléfono (9 o 10 dígitos) y para el email (formato de email correcto) en el caso de que el tipo de persona no sea HIJO ni HIJA; para este caso se solicita la fecha de nacimiento y se permite la inscripción en actividades.
3. El botón Añadir otra Persona guarda los datos introducidos y permite rellenar unos nuevos para tener otro miembro en la familia.

4. Datos necesarios para registrar un dato de contacto. La aplicación verifica que el usuario introduzca datos válidos para el teléfono (9 o 10 dígitos).
5. El botón Añadir otro Contacto guarda los datos introducidos y permite rellenar unos nuevos para tener otro dato de contacto en la familia.

6.3.3 Detalle Familia

A esta página se accede pulsando el botón “Ver Familia” en cualquiera de las tablas de listado de Familias.

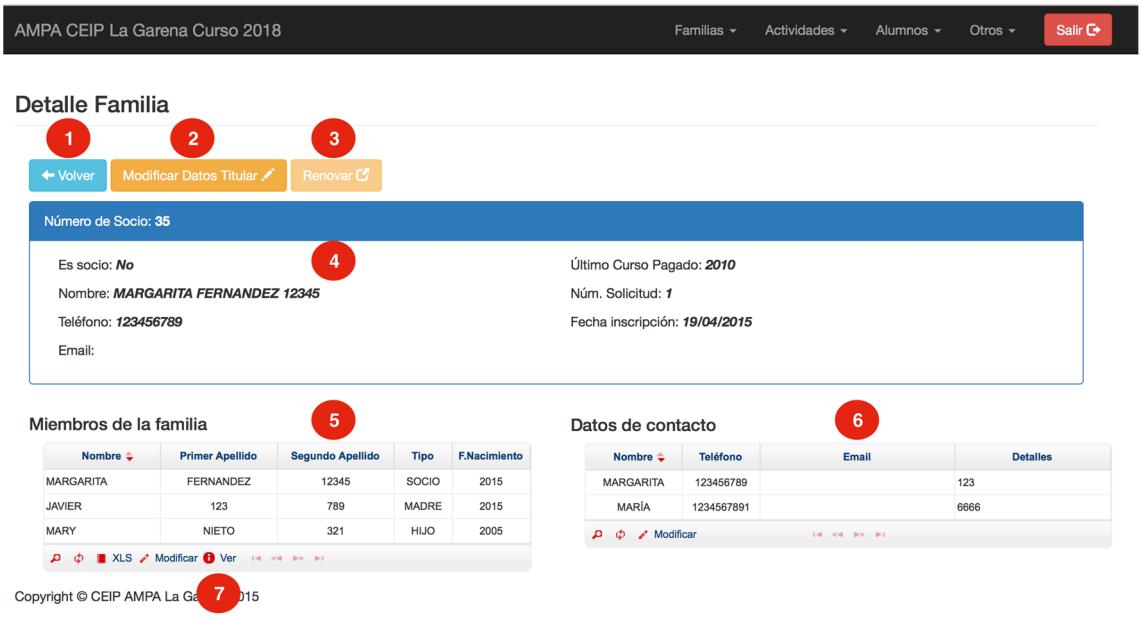

Ilustración 66 Manual de Usuario – Detalle Familia

1. Retorna a la página anterior con el listado de familias.
2. Navega a la página para editar los datos de la familia.
3. Direcciona a la página para renovar las actividades de todos los miembros de la familia. Esta opción sólo está habilitada cuando hay un curso posterior creado.
4. Panel de información con los datos de la familia.
5. Tabla con los miembros que forman parte de esa familia.
6. Tabla con los datos de contacto registrados para esa familia.
7. El botón “Ver” permite navegar a ver los datos de la persona seleccionada en la tabla.

6.3.4 Modificar Familia

A esta página se accede pulsando el botón “Ver Familia” en cualquiera de las tablas de listado de Familias y en la siguiente pantalla pulsando el botón “Modificar Datos Titular”. Muestra los datos para la familia seleccionada permitiendo su modificación.

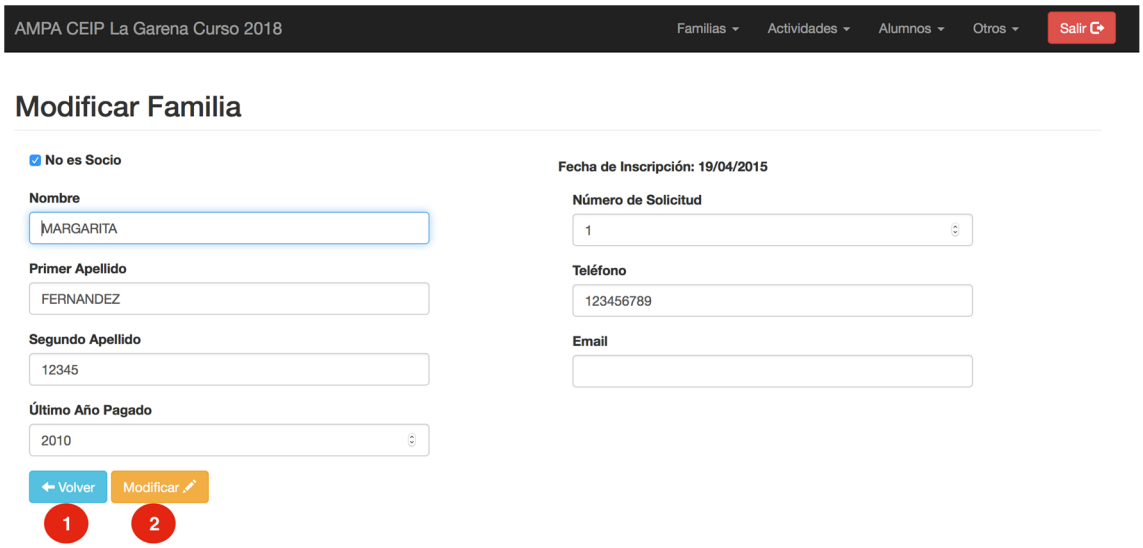

Ilustración 67 Manual de Usuario – Modificar Familia

1. Retorna a la página anterior con el detalle de la familia.
2. Verifica los datos del formulario comprobando que sean válidos, es decir, un teléfono con 9 o 10 dígitos, un email con formato correcto y un número de solicitud no existente en la base de datos para ese año; una vez verificados guarda los cambios en la base de datos.

6.3.5 Renovar Familia

A esta página se accede pulsando el botón “Ver Familia” en cualquiera de las tablas de listado de Familias y en la siguiente pantalla pulsando el botón “Renovar”. Muestra los datos para la familia seleccionada y todos sus miembros con las actividades en las que están registrados permitiendo la opción de renovar cualquiera de ellas.

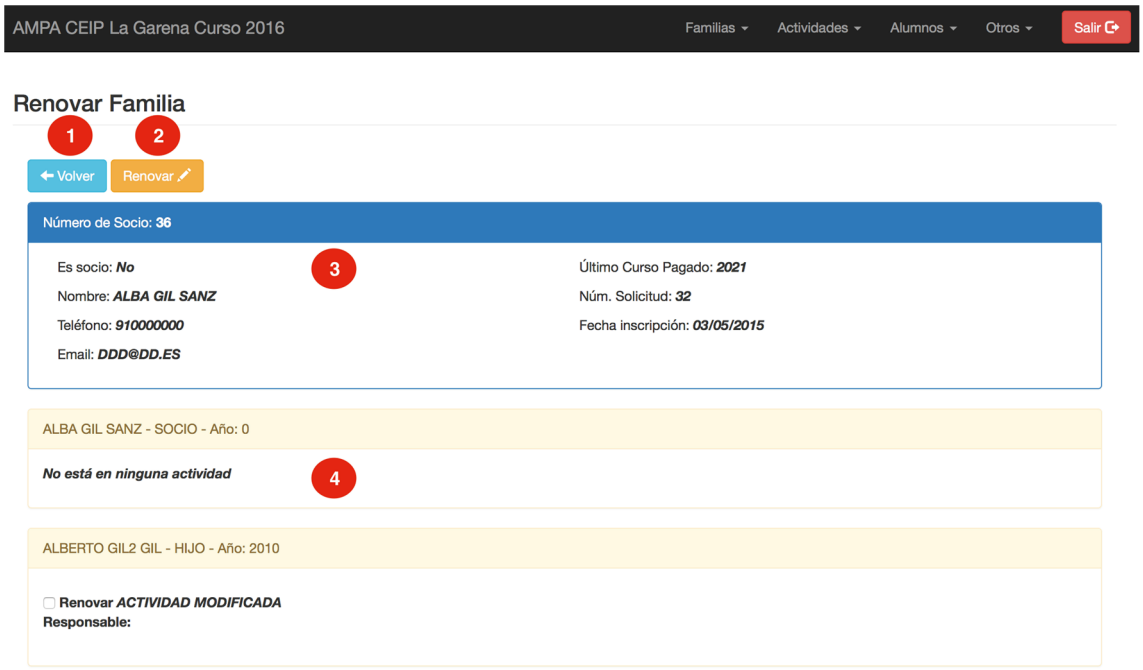

Ilustración 68 Manual de Usuario – Renovar Familia

1. Retorna a la página anterior con el detalle de la familia.
2. Botón que realiza la renovación de la familia.
3. Datos de la familia para la que se realiza la renovación.
4. Miembros de la familia y sus actividades a renovar.

6.3.6 Modificar Familiar

A esta página se accede pulsando el botón “Ver Familia” en cualquiera de las tablas de listado de Familias y en la siguiente pantalla pulsando el botón “Ver” en la tabla de miembros de la familia. Muestra los datos para la persona seleccionada permitiendo su modificación e incluso dando la posibilidad de inscribirle en actividades.

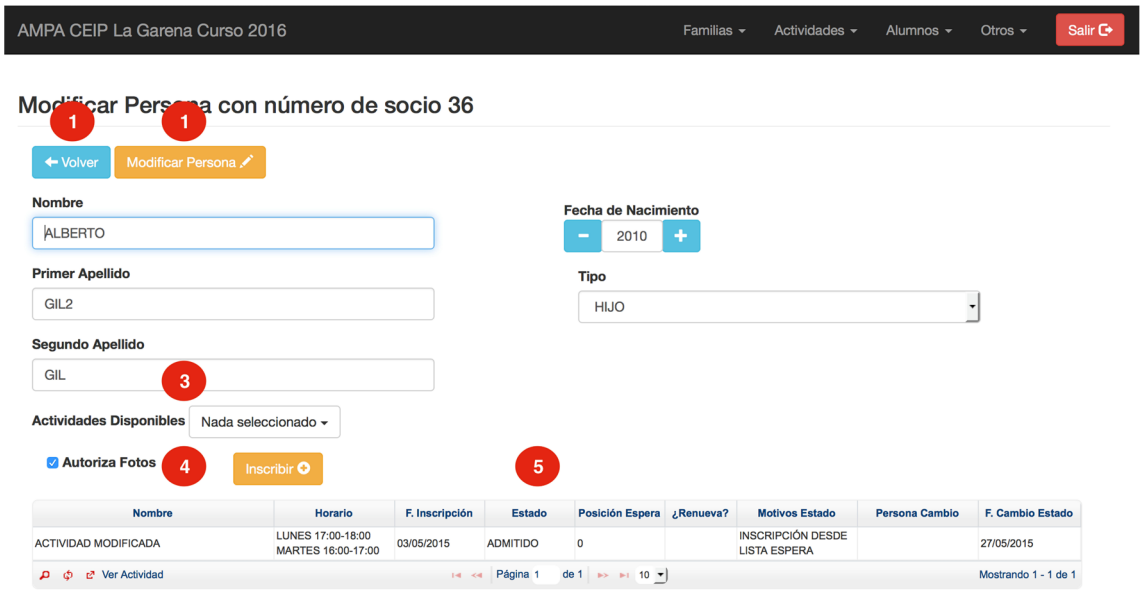

Ilustración 69 Manual de Usuario – Modificar Familiar

1. Retorna a la página anterior con el detalle de la familia.
2. Verifica los datos del formulario comprobando que sean válidos y guarda los cambios en la base de datos.
3. Actividades disponibles para ese miembro en las que se le puede inscribir.
4. Botón que realiza la inscripción en las actividades.
5. Tabla con las actividades en las que está registrado el usuario.

6.4 Menú Actividades

La sección del menú de Actividades cuenta con un total de 3 opciones: “Listado”, “Inscripciones” y “Añadir Actividad”. A continuación se explican todas las pantallas de este menú y aquellas que se derivan de ir accediendo a través de las distintas opciones.

6.4.1 Listado Actividades

Muestra un listado con todas las actividades que han sido registradas en la aplicación.

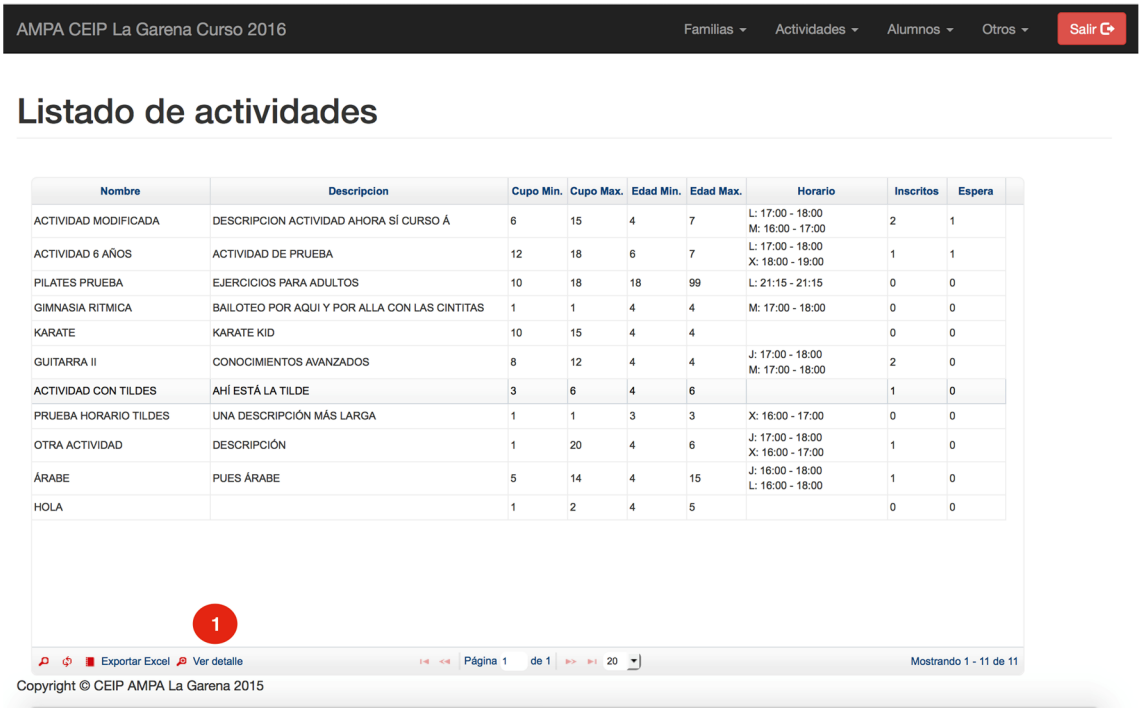

Ilustración 70 Manual de Usuario – Listado Actividades

1. El botón “Ver detalle” permite ir a la página que muestra la información de la actividad seleccionada.

6.4.2 Detalle Actividad

A esta página se accede pulsando el botón “Ver Detalle” en la tabla de listado de Actividades.

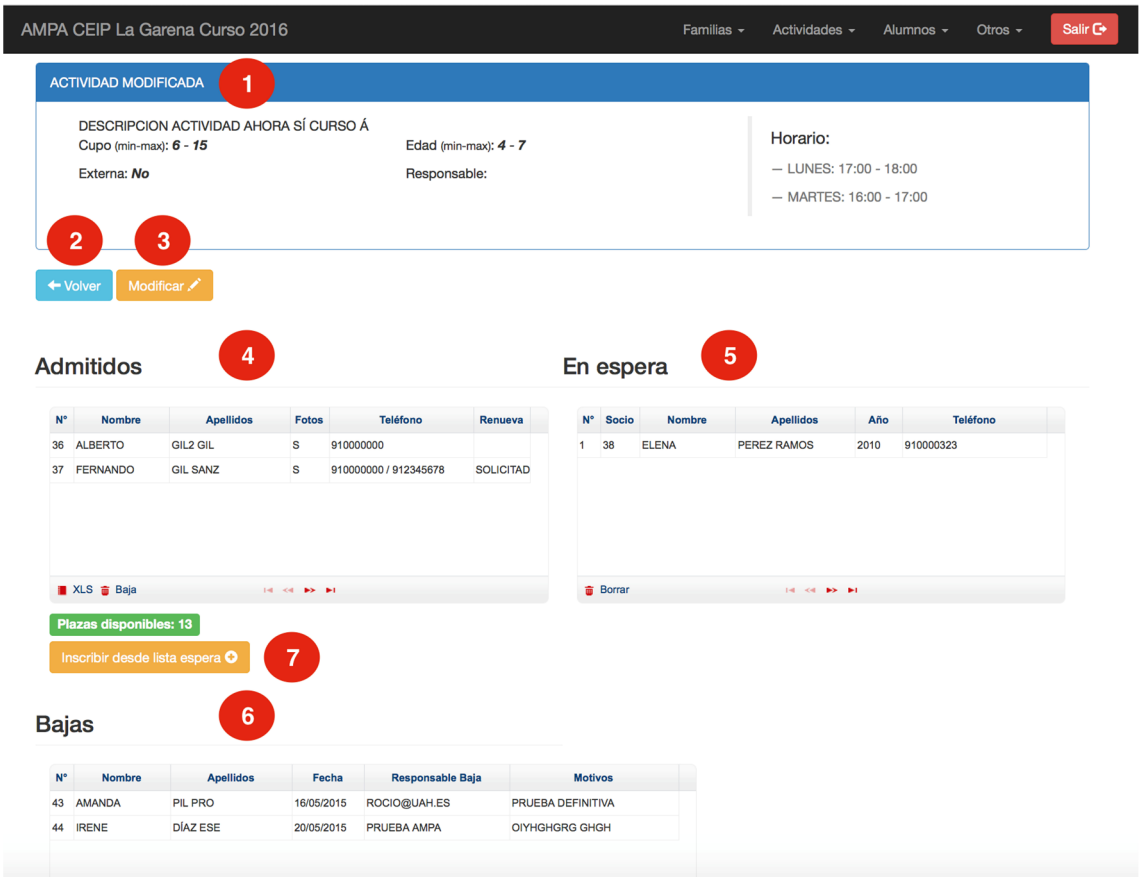

Ilustración 71 Manual de Usuario – Detalle Actividad

1. Panel con la información de la actividad.
2. Retorna a la página anterior con el listado de actividades.
3. Botón que navega a la página para modificar la actividad.
4. Tabla personas inscritas y admitidas en la actividad.
5. Tabla de personas inscritas pero que se encuentran en la cola de espera.
6. Tabla de personas que se han dado de baja de la actividad.
7. Botón que cambia a la primera persona de la tabla Espera a la tabla de Admitidos.

6.4.3 Modificar Actividad

A esta página se accede pulsando el botón “Ver Detalle” en la tabla de listado de Actividades y en la siguiente pantalla pulsando el botón “Modificar”. Muestra los datos para la actividad seleccionada permitiendo su modificación.

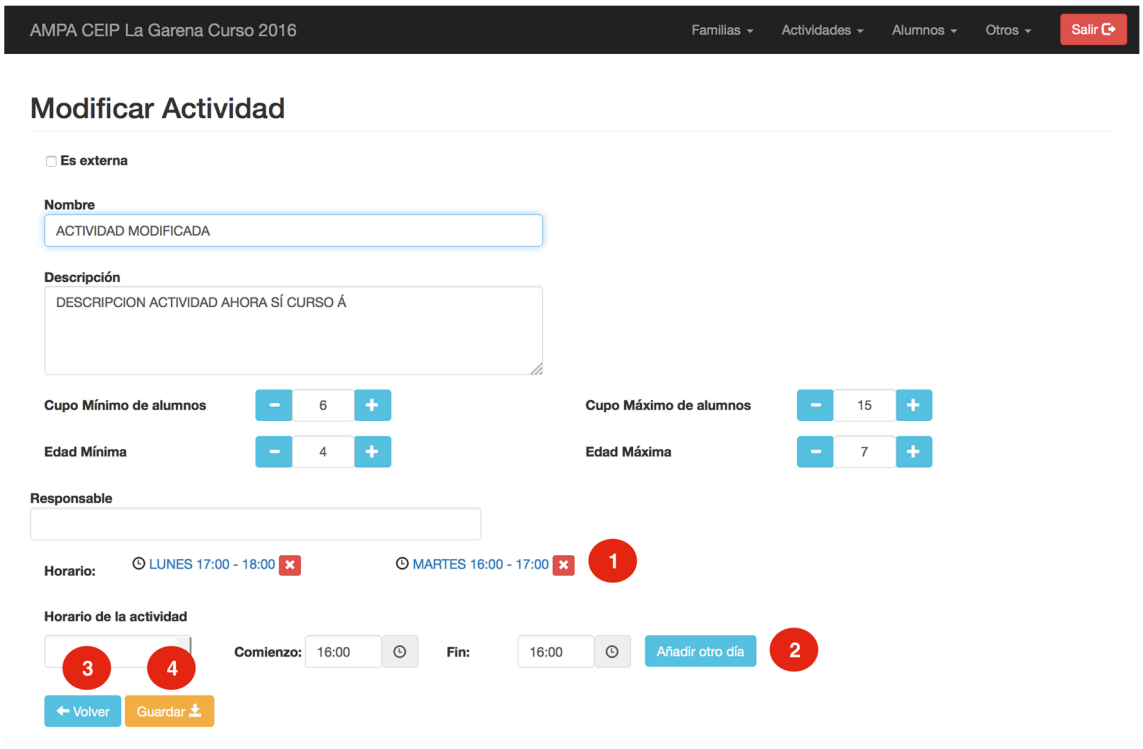

Ilustración 72 Manual de Usuario – Modificar Actividad

1. Horarios guardados hasta el momento para esa actividad. Se eliminan pulsando el botón con forma de aspa.
2. Controles para añadir un nuevo día en el que se impartirá la clase de la actividad.
3. Regresa a la página anterior.
4. Guardar los datos verificando primero que todos esté correctamente introducido.

6.4.4 Inscripciones

Está página permite realizar la inscripción de un miembro de una familia en alguna de las actividades registradas en la aplicación. Este procedimiento se realiza en tres pasos.

1. Se introduce nombre de la persona que queremos inscribir o bien un número de socio y se avanza al siguiente paso pulsando el botón Continuar.

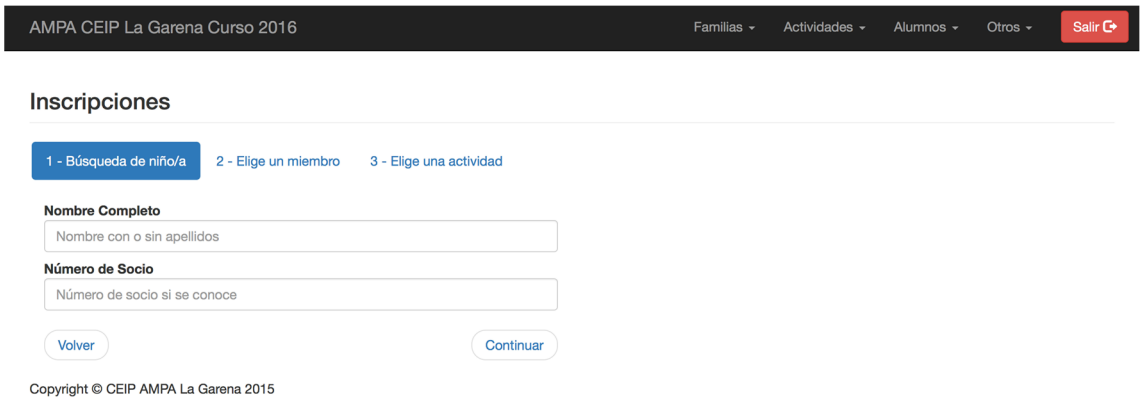

Ilustración 73 Manual de Usuario – Inscripciones Paso 1

2. La aplicación cambiará el formulario y ahora solicitará al usuario que elija una persona de entre las encontradas y que si dispone de un número de solicitud lo introduzca. Una vez rellenados estos datos se avanza pulsando el botón Continuar.

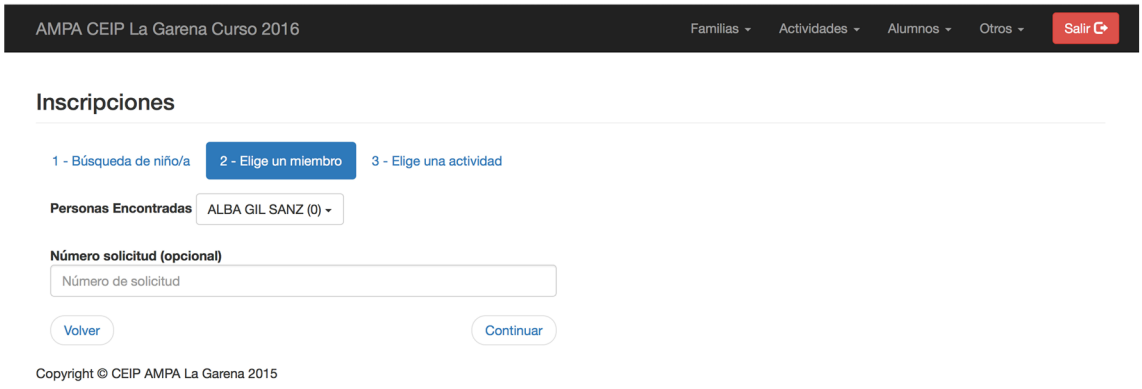

Ilustración 74 Manual de Usuario – Inscripciones Paso 2

3. En el nuevo formulario se elige de entre las opciones que presenta la aplicación, las actividades en las que se desea inscribir a la persona seleccionada. Se finaliza la inscripción pulsando el botón Finalizar.

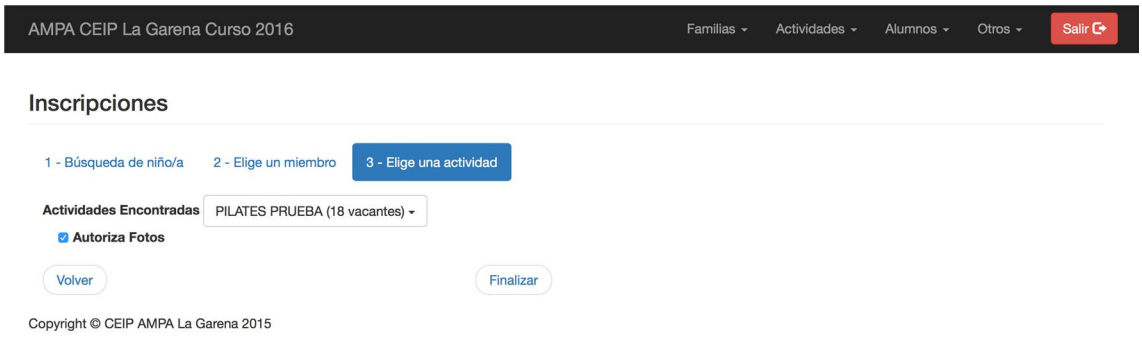

Ilustración 75 Manual de Usuario – Inscripciones Paso 3

6.4.5 Añadir Actividad

Cuando el usuario acceda a la opción “Añadir Actividad” se mostrará una página con un formulario solicitando la información básica necesaria para registrar una actividad en la aplicación.

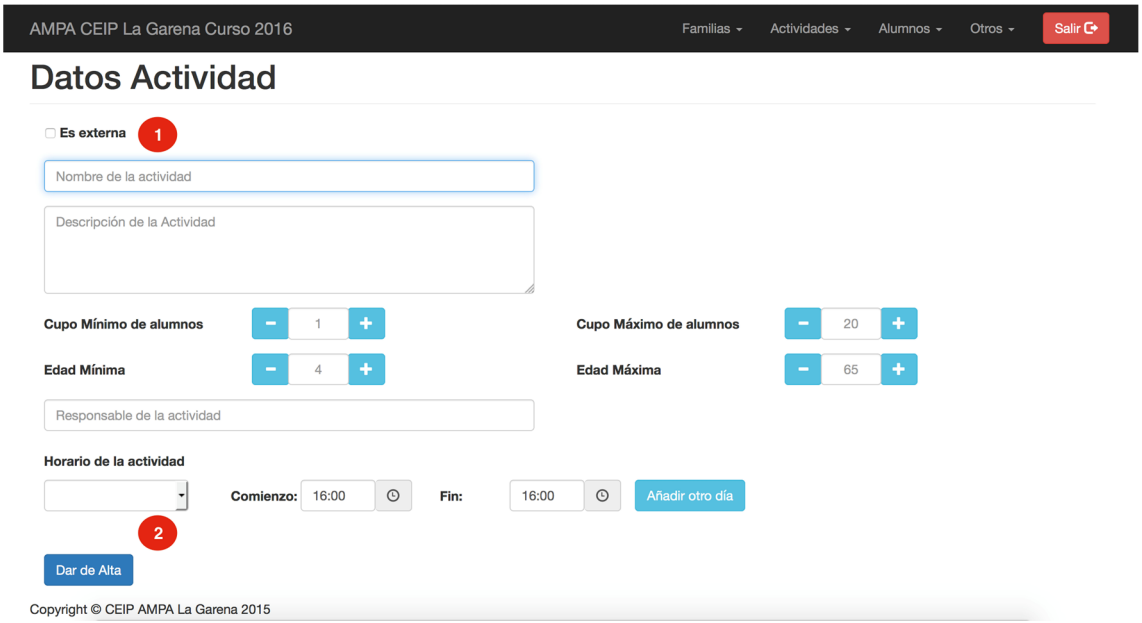

Ilustración 76 Manual de Usuario – Añadir Actividad

1. Datos necesarios para registrar una actividad. La aplicación verifica que el usuario introduzca datos válidos, es decir, solicita un nombre, un cupo mínimo y máximo de alumnos, una edad mínima y máxima para participar, un responsable y un horario para un día mínimo.
2. Botón para dar de alta la actividad.

6.5 Menú Alumnos

La sección del menú de Alumnos cuenta con un total de 3 opciones que son comunes: “Listado Completo”, “Listado Socios” y “Listado No Socios”. A continuación se explican todas las pantallas de este menú y aquellas que se derivan de ir accediendo a través de las distintas opciones.

6.5.1 Listado Completo, de Socios de No Socios

Estas tres opciones comunes y llevan a la misma página que es un listado de los alumnos que hay en la aplicación. Si se elige la opción “Socios” sólo se mostrará un listado con los alumnos que son socios, si se elige la opción “No Socios” sólo se mostrará un listado con los alumnos que no son socios y si por el contrario se elige la opción “Listado Completo” se mostrará un listado con todos los alumnos que han sido registrados en la aplicación.

Ilustración 77 Manual de Usuario – Listado Alumnos

1. El botón “Ver Alumno” permite ir a la página que muestra la información del alumno seleccionado.

6.5.2 Ver Alumno

A esta página se accede pulsando el botón “Ver Alumno” en cualquiera de las tablas de con listado de Familia y en la siguiente pantalla pulsando el botón “Ver” en la tabla de miembros de la familia. Muestra los datos para la persona seleccionada permitiendo su modificación e incluso dando la posibilidad de inscribirle en actividades.

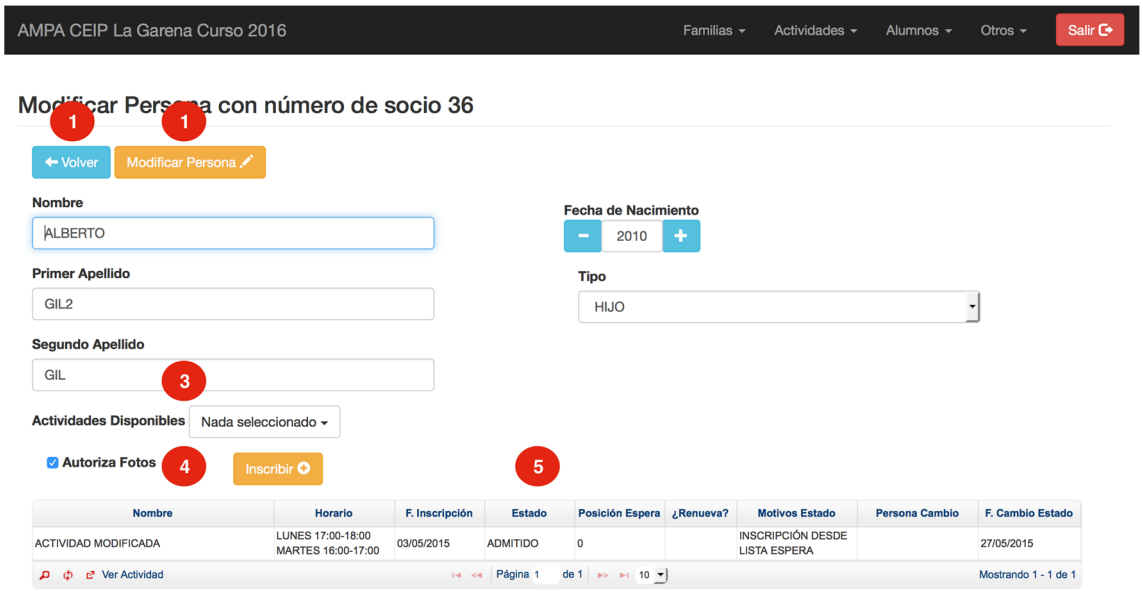

Ilustración 78 Manual de Usuario – Ver Alumno

1. Retorna a la página anterior con el listado de alumnos.
2. Verifica los datos del formulario comprobando que sean válidos y guarda los cambios en la base de datos.
3. Actividades disponibles para ese miembro en las que se le puede inscribir.
4. Botón que realiza la inscripción en las actividades.
5. Tabla con las actividades en las que está registrado el usuario.

6.6 Menú Otros

La sección del menú de otros cuenta con 2 opciones, Cursos y Generar Circular.

6.6.1 Cursos

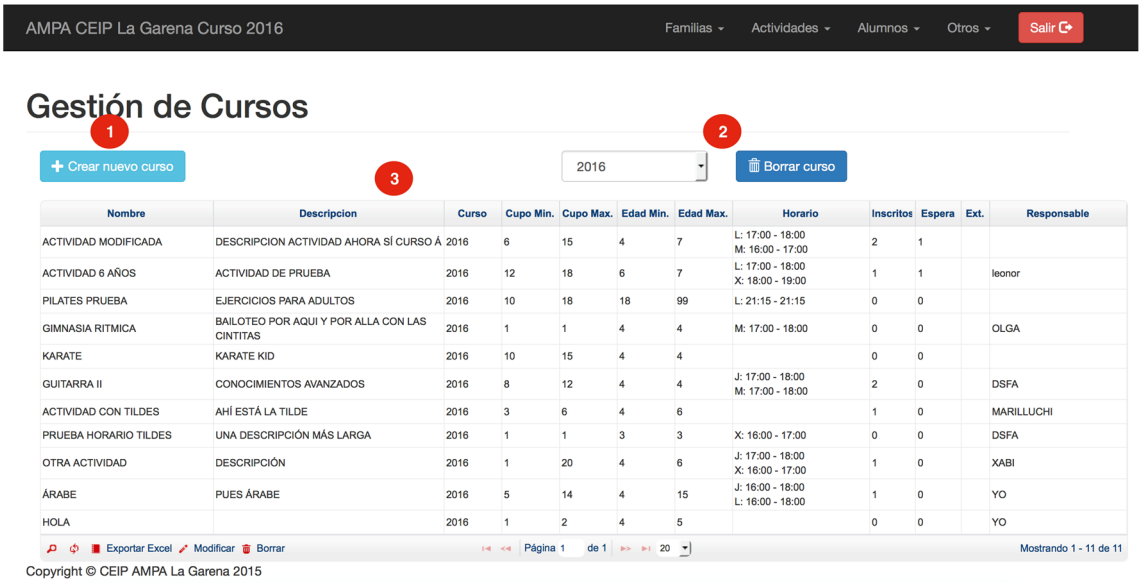

Ilustración 79 Manual de Usuario – Gestión de Cursos

1. Permite la creación de un nuevo curso con las actividades del curso actual.
2. Despegable para elegir un curso y el botón correspondiente para proceder a su eliminación.
3. Tabla con la información de las actividades para el curso actual elegido en la pantalla principal.

6.6.2 Generar Circular

Está página permite enviar un email idéntico a todas las familias o bien sólo a las familias socias registradas en la aplicación.

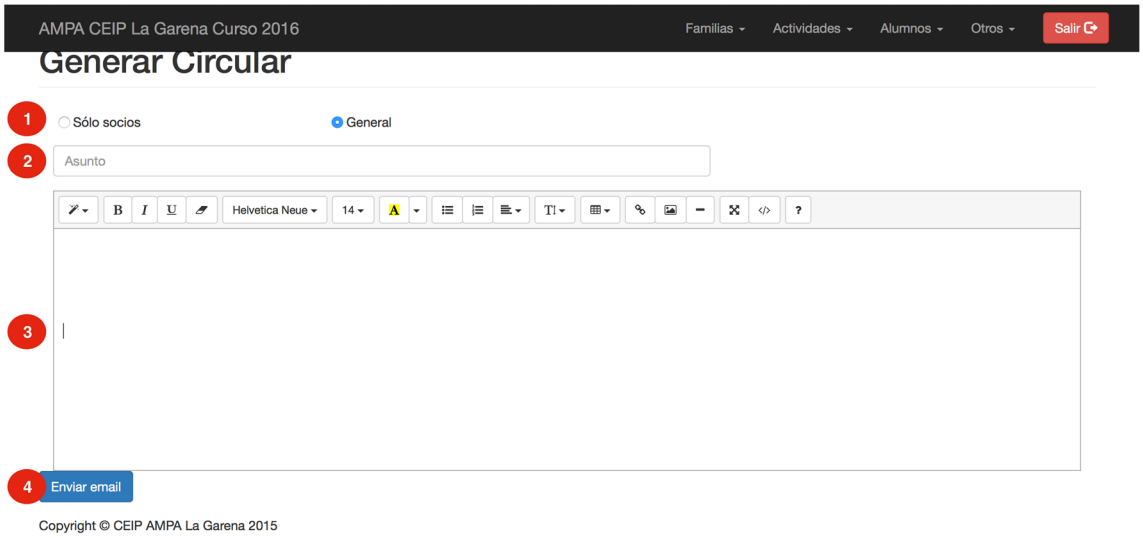

Ilustración 80 Manual de Usuario – Enviar Circular

1. Para elegir los destinatarios del email.
2. Para introducir el asunto del email.
3. Cuadro mando para redactar el email.
4. Botón que realiza en envío del email.

Capítulo 7
7 Contenido del CD

Adjunto a esta memoria existe un CD que contiene todos los programas y ficheros necesarios para poder desplegar el proyecto en cualquier PC tanto para su puesta en funcionamiento como para continuar desarrollando su código. A continuación se desglosa las carpetas y su contenido incluido.

- **32 Bit**

  Contiene el fichero de instalación del JDK de JAVA para la versión de 32 bits
- **64 Bit**

  Contiene el fichero de instalación del JDK de JAVA para la versión de 32 bits
- Código Fuente

  Incluye la carpeta con el código fuente al completo preparado para ser importado desde Eclipse y además un fichero WAR para desplegar la aplicación y ponerla en funcionamiento.
- **MySQL**

  Contiene dos ficheros: un instalador de MySQL y el fichero script para la creación de la base de datos necesaria para funcionar con el proyecto con todas sus tablas, funciones y procedimientos almacenados.
- **Programas**

  Incluye el fichero de instalación de Apache Tomcat 8 y el fichero comprimido con el Eclipse en su versión Luna para Web Developers.

Capítulo 8
8 Bibliografía

Dynamic Reports. (s.f.). *Dynamic Reports Documentation*. Obtenido de http://www.dynamicreports.org/documentation/documentation-overview

Oracle JAVA Documentation. (s.f.). *Interface Filter*. Obtenido de http://docs.oracle.com/javaee/5/api/javax/servlet/Filter.html

Oracle JSP Documentation. (s.f.). *The JAVA EE 5 Tutorial*. Obtenido de https://docs.oracle.com/javaee/5/tutorial/doc/bnajo.html

Wikipedia. (s.f.). *DAO*. Obtenido de http://es.wikipedia.org/wiki/Data_Access_Object

Wikipedia. (s.f.). *Desarrollo en cascada*. Obtenido de http://es.wikipedia.org/wiki/Desarrollo_en_cascada

Wikipedia. (s.f.). *HTML*. Obtenido de http://es.wikipedia.org/wiki/HTML

Wikipedia. (s.f.). *HTML5*. Obtenido de http://es.wikipedia.org/wiki/HTML5

Wikipedia. (s.f.). *Java (lenguaje de programación)*. Obtenido de http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

Wikipedia. (s.f.). *Java Servlet*. Obtenido de http://es.wikipedia.org/wiki/Java_Servlet

Wikipedia. (s.f.). *JavaScript*. Obtenido de http://es.wikipedia.org/wiki/JavaScript

Wikipedia. (s.f.). *JavaServer Pages*. Obtenido de http://es.wikipedia.org/wiki/JavaServer_Pages

Wikipedia. (s.f.). *JDBC*. Obtenido de http://es.wikipedia.org/wiki/Java_Database_Connectivity

Wikipedia. (s.f.). *jQuery*. Obtenido de http://es.wikipedia.org/wiki/JQuery

Wikipedia. (s.f.). *Modelo MVC*. Obtenido de http://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

Wikipedia. (s.f.). *Modelo Relacional*. Obtenido de http://es.wikipedia.org/wiki/Modelo_relacional

Wikipedia. (s.f.). *MySQL*. Obtenido de http://es.wikipedia.org/wiki/MySQL

Wikipedia. (s.f.). *Programación Orientada a Objetos*. Obtenido de http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

Wikipedia. (s.f.). *SMTP*. Obtenido de http://es.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

Wikipedia. (s.f.). *Tomcat*. Obtenido de http://es.wikipedia.org/wiki/Tomcat

Wikipedia. (s.f.). *Twitter Bootstrap*. Obtenido de http://es.wikipedia.org/wiki/Twitter_Bootstrap

Capítulo 9
9 Índice de Figuras y Tablas

9.1 Figuras

9.2 Tablas

Universidad de Alcalá
Escuela Técnica Superior de Ingeniería Informática

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá